195 PTAS.
(IVA Incluido)
mi computer
114
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.
P.V.P. Canarias

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen X-Fascículo 114

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 120 fascículos de aparición semanal, encuadernables en diez volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-183-6 (tomo 10)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 258603
Impreso en España-Printed in Spain- Marzo 1986

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (120 fascículos más las tapas, guardas y transferibles para la confección de los 10 volúmenes) son las siguientes:

a) Un pago único anticipado de 27 105 ptas. o bien 10 pagos trimestrales anticipados y consecutivos de 2 711 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

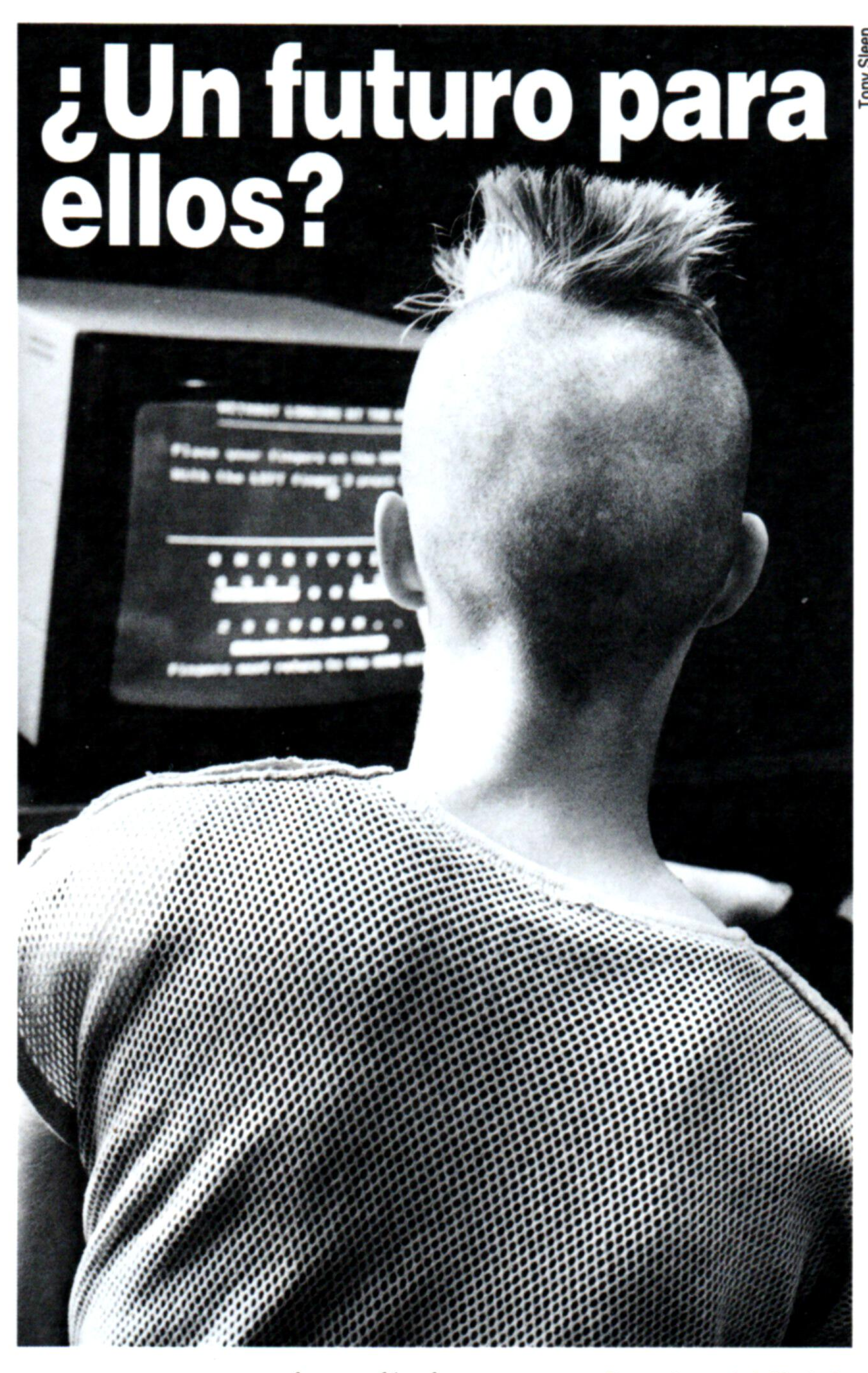
Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.



En este primer capítulo de esta nueva serie, analizaremos las aptitudes que se requieren para desenvolverse en el campo laboral de la informática

¿Un futuro para ellos?



Tony Sleep

¿Perspectivas más brillantes?
La industria informática ofrece una gran cantidad de posibilidades para quienes buscan trabajo, pero es poco probable que usted consiga un puesto de trabajo contando sólo con experiencia en ordenadores personales. Una formación profesional en un centro educativo cualificado representa un buen comienzo para una carrera que, aunque inicialmente sea variada, a menudo implicará una especialización en un campo determinado. En casos extremos, esto puede conducir a perspectivas de empleo más reducidas en el futuro, a menos que se haga un serio esfuerzo para mantenerse al día en cuanto a innovaciones de hardware y software

La primera decisión que ha de adoptar el postulante a un puesto de trabajo en el medio informático es en qué parte de esta compleja industria desea trabajar, teniendo presente que los buenos trabajos escasean en todas partes. En esta serie consideraremos carreras que difieren tanto entre sí como el día de la noche, ofreciendo cada una de ellas una gratificación laboral diferente: sueldos, perspectivas de futuro, compromiso técnico y satisfacción personal.

Un punto a considerar es que normalmente una carrera de informática sigue un rumbo fijo. Si usted decide avanzar en una dirección, entonces ése es el camino en el cual seguirá durante cierto tiempo. Esto es particularmente cierto si la firma que lo ha contratado realiza una fuerte inversión en su formación profesional o si usted obtiene conocimientos y destreza especializados.

En informática la especialización laboral es muy amplia, aunque hay algunas categorías comunes. Cada trabajo plantea exigencias diferentes de personal y a su vez ofrece diferentes incentivos y perspectivas. Los fabricantes de ordenadores y electrónica, por ejemplo, requieren tres tipos de personal, todos ellos trabajando en hardware, y lo mismo puede decirse de las telecomunicaciones, un área de la industria que cada vez adquiere mayor relevancia. Existen trabajadores de línea de producción como los de cualquier otra fábrica, con salarios algo superiores pero similares perspectivas; ingenieros de producción, cuya tarea es asegurar que el producto final funcione, y personal de investigación y diseño.

Diseño de ordenadores

El diseño de ordenadores, ya sea de microprocesadores o de sistemas completos, es una especialidad destinada a quienes poseen una relativa experiencia (o para auténticos genios), pero la *ingeniería* también está abierta a quienes poseen un talento medio. Ambos trabajos requieren personal entrenado. Los proveedores que han obtenido éxito con sus productos tienden a emplear más personal para desarrollar la siguiente gama de éstos.

Estos trabajos de diseño son territorio exclusivo para graduados, quienes exigen salarios elevados, seguridad y prestigio. En la industria los graduados son más numerosos que los no graduados en una proporción de nueve a uno, según el organismo británico COSIT (*Computing services industry training council*: consejo de adiestramiento de la industria de servicios informáticos). Un curso académico que se aconseja seguir es uno de ingeniería electrónica o informática en una universidad o instituto politécnico. Exigiendo niveles básico y avanzado, en tres años se enseña al alumno teoría básica y práctica avanzada.

Los niveles avanzados exclusivamente en muy raras ocasiones son suficientes: los patrones esperan que uno sea capaz de arreglárselas con una placa de circuito impreso, además de conocer los principios sobre los que opera un microordenador.

Es probable que ni siquiera los programadores más brillantes ni los aficionados a la electrónica empiecen diseñando un PC o un chip. Hay otras aptitudes que son esenciales antes de que se le permita diseñar siquiera una pieza pequeña. Un proyecto de diseño por lo general se comparte entre un equipo formado por una docena o incluso varios centenares de trabajadores, desarrollando y completando subsecciones del proyecto completo.

Los patrones tienden a pensar que la experiencia académica es una ventaja en informática y electrónica porque permite que los graduados se adapten a la velocidad a la cual las nuevas tecnologías se aplican a los productos de hoy en día. Existe, sin embargo, la desafortunada creencia de que las mujeres son incapaces de adecuarse a los cambios:

Personal de ventas al detalle

Un número creciente de jóvenes se inicia en el mundo laboral trabajando en una tienda de ordenadores o en una casa de sistemas con puertas a la calle a través de la que se venden al público pequeños sistemas, periféricos, suministros y software. Las ventas en las tiendas no suelen ser frecuentes, y su relativa importancia respecto a otras vías de comercialización las sitúa en el extremo inferior del mercado.

El trabajo permite que el recién llegado aprenda sobre ordenadores personales, pero pocas veces le permite familiarizarse con una amplia gama de sistemas o técnicas de programación. No obstante, para algunas personas es un punto de entrada válido hacia una carrera en el campo de las ventas y marketing con una firma importante. Tales trabajos no suelen estar bien remunerados y exigen cumplir largas jornadas

la proporción de hombres y mujeres en la industria es de cuatro a uno a favor de los primeros (cifras del COSIT). Afortunadamente, este errado concepto está siendo cuestionado con mayor fuerza cada día.

Sin embargo, los entusiastas pueden convertirse en personal de ingeniería con poco o nada de entrenamiento formal. Siempre se solicita personal que sepa instalar una pieza de equipo electrónico, pero el entrenamiento y el salario son limitados y las perspectivas para el futuro son poco halagüeñas. Las posibilidades por lo general se limitan a las firmas de microordenadores más pequeñas.

En la industria del ordenador, el software y el mantenimiento son las principales fuentes de empleo. Los programadores (definidos de forma genérica de modo de incluir a otro sector de personal de apoyo) representan más de la mitad del personal; el personal de ventas y marketing constituye la tercera parte.

La mayoría de los programadores continúan trabajando ya sea en operaciones de proceso de datos (*data processing*: DP) —que ahora tienden a desarrollarse en servicios de información de administración (*management information services*: MIS)— o bien con proveedores de ordenadores, casas de software o agencias de informática.

El DP constituye la vía más utilizada para iniciar una carrera en informática, pero está disminuyendo en importancia a medida que la operatoria de los ordenadores se vuelve más fácil y más automática.

Ciertas categorías laborales de bajo nivel, como operatoria, programación o análisis de sistemas básicos, quizá se puedan considerar como un paso en la carrera, no como un fin en sí mismas.

En la actualidad la programación es un tema sumamente especificado. Es poco probable que un experto en un lenguaje de microordenador como el FORTH o el PASCAL pueda captar de inmediato las particularidades más sutiles del COBOL o el FORTRAN en un ordenador central, y la situación se complica por el predominio del PASCAL y el COBOL en las áreas de gestión, mientras que el FORTH y el FORTRAN son más populares en las áreas técnicas. El C también está adquiriendo cada vez mayor importancia, en particular en lo que se refiere a la programación de sistemas.

Los sueldos de los informáticos

Categoría	Septiembre 1984	Marzo 1985
Director de informática	6.6	8.9
Director de DP	9.6	8.6
Analista de sist. senior	9.0	10.5
Analista senior	7.3	7.9
Programador senior	4.8	8.6
Programador analista	6.5	7.8
Programador	5.7	1.6
Ejecutivos	**6.6**	**7.6**

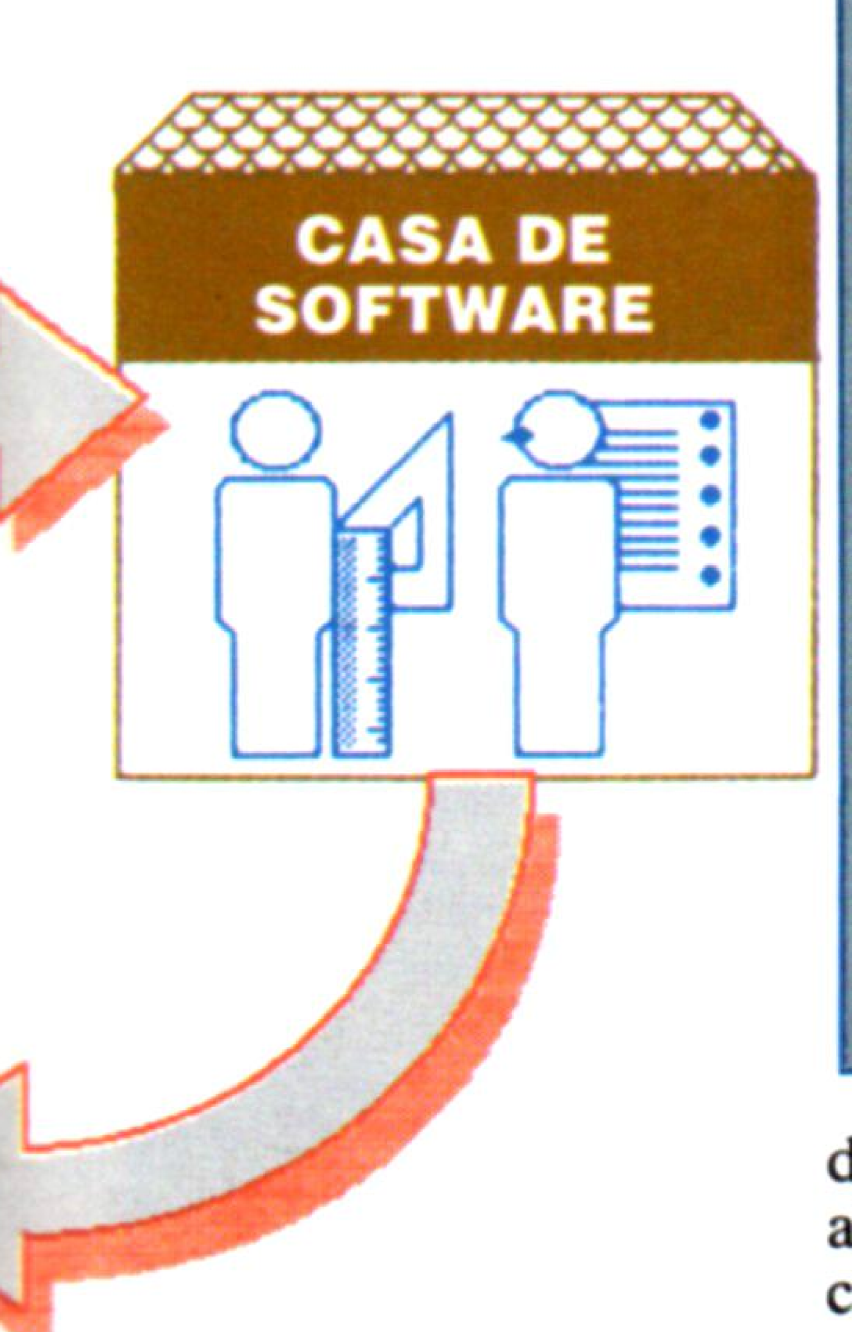
Estructura salarial
En esta tabla se refleja la cambiante tabla salarial para una gama de personal de ordenadores en Gran Bretaña. Las cifras, que representan algunos cambios notables en poco más de seis meses, están sacadas del Regional Reward Survey, informe dado a conocer en septiembre de 1985. Las cifras para los aumentos salariales anuales están promediadas a través de todo el país, y se deben comparar con la inflación, que durante el mismo período fue de varios puntos

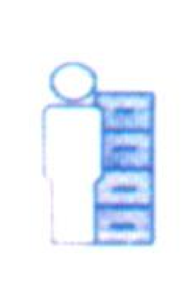
Una proporción significativa de las personas de la industria del ordenador (quizá una tercera parte del personal) trabaja para suministradores de ordenadores. Además de la fabricación, el diseño y el apoyo de mantenimiento hay muchísimo personal de ventas. Éste puede trabajar directamente tanto para un proveedor de ordenadores como para un



Ingenieros de mantenimiento

Un ingeniero de mantenimiento requiere conocimientos generales de ingeniería eléctrica y un conocimiento básico de programación. El sueldo y las condiciones son buenos, en compensación por el sacrificio que supone el tener que trabajar a través de una vasta zona geográfica y tener que estar disponible las 24 horas del día. Los ingenieros con base en la fábrica tienen una tarea más sencilla, descubriendo y arreglando fallos en las máquinas que les traen o que se hallan en proceso de fabricación. La tarea que realiza el ingeniero de mantenimiento exige, sobre todo, una gran paciencia: hallar un chip defectuoso en una habitación llena de ordenadores centrales es agotador. Un conocimiento básico de programación ayuda al ingeniero a rastrear un fallo en un conjunto de placas de circuito impreso o incluso en una placa individual. Este tipo de trabajo tenderá cada vez más a desaparecer en la industria a medida que los ordenadores vayan siendo más fiables y los sistemas de diagnóstico en línea se encarguen de buscar los errores. En el futuro cercano el ingeniero de mantenimiento de campo podría ver limitado su trabajo a entregar una pieza de recambio, siendo el propio ordenador el que localice su avería y solicite el recambio adecuado







Caroline Clayton

Relaciones industriales
El hardware continúa siendo el centro de la industria del ordenador, pero quien busque empleo quizá encuentre una fuente más segura especializándose en una de las actividades (marketing, p. ej.) que giran alrededor del mismo. El diagrama ilustra las diversas relaciones existentes entre el fabricante, el minorista, el fabricante independiente y el usuario final

distribuidor, una casa de sistemas o de software, o alguno de los muchos comercios minoristas de microordenadores.

Al igual que en otros sectores, la industria del ordenador divide al personal de ventas en tres categorías. En primer lugar está el personal de venta "directa", si bien los que "viajan" recogiendo pedidos directos constituyen una novedad relativa. A la mayoría de ellos se les encarga el manejo de varias cuentas.

En segundo lugar está el personal de marketing, cuya labor consiste en generar contactos con compradores de ordenadores ya existentes (en el caso de grandes sistemas) o, más raramente, con compradores noveles. Una empresa de ordenadores centrales como IBM entrena a estos empleados de marketing para que sean el verdadero personal de ventas, hablando directamente con el gerente de DP/MIS, el director gerente o la junta del potencial cliente. Un proveedor de microordenadores pequeño contratará un director de marketing para asegurar que los distribuidores y minoristas tengan existencias de sus productos.

El tercer trabajo de ventas implica unir entre sí los diversos elementos del aprovisionamiento y apoyo de hardware, software y sistemas. En la industria del ordenador, los papeles de administración como éste están todavía en manos de ex programadores o ingenieros.

El adiestramiento se encuentra inmerso en un laberinto de siglas; por este motivo es esencial, antes de iniciar un curso, averiguar lo que proporcionará cada conjunto de iniciales. En Gran Bretaña, por ejemplo, país muy avanzado en este campo, es la Manpower Services Commission (MSC: comisión de servicios de recursos humanos) el organismo que se encarga de la formación profesional y de impartir cursos básicamente sobre programación y

Escribir juegos para ordenador

La era del adolescente millonario escritor de juegos para ordenador ha terminado (si es que había llegado a iniciarse). La idea de que cualquiera podía escribir un juego *best-seller* que, como un disco de música *pop*, vendiera millones de copias de la noche a la mañana, no estuvo sustentada más que por un puñado de ejemplos cuestionables.

Los juegos los escriben una cantidad significativa de programadores independientes, muchos de los cuales, siendo muy jóvenes, se introdujeron en el negocio al aparecer ordenadores personales baratos y potentes y haber una gran demanda de software empaquetado, incluyendo juegos. Son pocos los programadores que trabajan de forma aislada. La mayoría de ellos trabajan para empresas pequeñas bajo contrato o venden a una empresa los derechos para fabricar, comercializar y distribuir sus programas. Una carrera de características tan independientes plantea problemas importantes. En primer lugar, es preciso trabajar todos los días, tanto si se tienen ganas como si no, para prosperar. Muchos programadores independientes tienen otro empleo en relación de dependencia y trabajan como independientes sólo a tiempo parcial. En segundo lugar, ha habido un alarmante número de quiebras entre pequeñas casas de software dedicadas a los mercados de juegos y ordenadores personales. Por último, escribir juegos no es, en realidad, una carrera. No hay nadie que enseñe a programar. Existe una tendencia a estancarse y retornar a lo que uno hace mejor y, por tanto, a no desarrollar aptitudes nuevas

técnicas de software bajo el auspicio del Training Opportunities Program (TOPS: programa de oportunidades de adiestramiento), diseñado en gran parte para el reciclaje profesional.

El TOPS ha sido criticado por su lentitud para actualizar sus cursos de modo de incluir métodos modernos, pero al estar sobre la mesa las propuestas del comité Alvey del Department of Trade and Industry (Departamento de Comercio e Industria) para corregir este defecto, parece ser que seguirá siendo la principal fuente de personal para programación de ordenadores después de las universidades y los politécnicos.

Los ITEC (Information Technology Education Centres: centros de educación en tecnología de la información) constituyen otro plan de captación, habiéndose creado más de 150 centros desde 1982 a lo largo y ancho de las islas británicas para proporcionar adiestramiento a los jóvenes que se encuentran en la encrucijada de terminar la escuela y conseguir un primer empleo. Bajo el YTS (Youth Training Scheme: plan de adiestramiento juvenil) se remunera a las personas de entre 16 y 19 años de edad que asistan a las clases de formación.

Mientras tanto, los fabricantes de ordenadores siguen estando entre quienes ofrecen los más altos salarios, y se les considera los mejores patrones en cuanto a condiciones generales, formación y perspectivas para el futuro que ofrecen. Por lo general esto es aplicable a los distribuidores y a las casas de software, pero es exactamente lo contrario en el caso de los comercios minoristas de ordenadores.

Lo pequeño es bello

Iniciamos esta serie dedicada al estudio del sistema operativo Unix considerando su historia y la filosofía de su diseño

Además de funciones básicas de E/S y facilidades para gestión de archivos y memoria, los sistemas operativos proporcionan un juego de *herramientas* o *utilidades* que crean un entorno de trabajo adecuado para el usuario. Estas utilidades pueden ir desde compiladores para diversos lenguajes hasta editores de texto y formateadores de impresión. De hecho, los principales puntos fuertes del Unix, además del hecho de ser un sistema operativo simple y a la vez potente y elegante, son la gran variedad de herramientas que proporciona y el modo en que permite utilizar la salida de un programa como entrada para otro. Al hacerlo, permite conectar entre sí varias herramientas simples para llevar a cabo las tareas más complicadas. La filosofía del Unix se puede enunciar con las siguientes máximas:

1. Escribir programas pequeños y simples que realicen bien una tarea.
2. Esperar utilizar la salida de cualquier programa como la entrada para otro, aun cuando el segundo programa no se haya escrito todavía.
3. Escribir siempre programas nuevos para realizar tareas nuevas, en vez de modificar los antiguos; de ese modo usted acrecienta la biblioteca de herramientas disponibles.

Hay, sin embargo, una dificultad. La idea de contar con una amplia variedad de herramientas que se pueden interconectar para llevar a cabo tareas complejas es lógica para los programadores y otros profesionales de la informática, pero no hace que el sistema sea muy "amable" con los usuarios inexpertos. Si usted desea hacer un uso exhaustivo de un sistema Unix, son muchas las cosas que ha de recordar.

La única forma de superar esto es proporcionar un sistema que puedan utilizar las personas sin experiencia: una "fachada" amable con el usuario que le esconda la complejidad y potencia completa del Unix. No obstante, es virtualmente imposible hacer esto de modo tal que la potencia permanezca disponible en su totalidad; por este motivo, sistemas Unix se encuentran fundamentalmente en instituciones académicas antes que en sistemas de gestión o personales.

El Unix lo desarrolló casi por accidente el programador Ken Thompson cuando trabajaba en los Laboratorios Bell, que constituye el núcleo de investigación de la gigantesca AT&T Corporation norteamericana. Thompson se encontraba realizando una simulación del movimiento planetario en un ordenador GE645, que estaba ejecutando uno de los primeros sistemas operativos multiusuarios denominado Multics. Los problemas propios de desarrollar software utilizando un entorno hostil en un gran ordenador lo llevaron a transferir sus esfuerzos a un pequeño ordenador DEC, un PDP-7. Para hacerlo, escribió un OS que daba la mayoría de las facilidades del Multics en la máquina más pequeña. Tuvo tanto éxito que atrajo la atención de Dennis Ritchie y otros colaboradores, quienes en 1971, en los mismos laboratorios, desarrollaron el sistema convirtiéndolo en la primera versión totalmente operativa de Unix.

Al igual que la mayoría de los sistemas operativos de la época, el Unix se escribió primero en ensamblador, y operó adecuadamente sólo en el miniordenador DEC, fundamentalmente en la serie PDP-11. Sin embargo, Dennis Ritchie también estaba trabajando en un lenguaje llamado B, un desarrollo del BCPL, el lenguaje de alto nivel que ofrecía casi el mismo control directo sobre el hardware que el ensamblador. Enseguida éste se convirtió en el C y el Unix se reescribió en este nuevo lenguaje. Sólo una pequeña fracción del Unix continúa estando escrita en ensamblador, lo que facilita la transferencia del código a máquinas diferentes: todo cuanto se necesita es un compilador de C. En la actualidad el Unix se ejecuta virtualmente en cualquier tipo de máquina, desde los mayores ordenadores centrales hasta los micros de 16 bits, y algunas versiones limitadas de Unix incluso se pueden ejecutar en micros de ocho bits.

Hasta hace poco tiempo era bastante difícil y costoso adquirir el Unix para uso comercial, pero AT&T tiene la política de conceder licencias casi gratuitas para las instituciones académicas, de modo que el uso del Unix se ha generalizado en facultades y universidades. En el mundo comercial se ha abierto camino especialmente en microorde-



PC PAM
Aunque el Unix por lo general se suele encontrar en miniordenadores, Hewlett-Packard ha implementado una versión, llamada HP-UX, para ejecutar en su Integral PC. Este potente portátil multitareas incluye una pantalla de plasma, impresora integral de chorro de tinta y 768 Kbytes de memoria, y opera bajo un sistema WIMP denominado PAM

Liz Heaney

La historia del Unix

Al haber sido desarrollado por un pequeño equipo, el Unix continúa siendo sensible y compacto. Muchas de sus excelentes facilidades se han abierto camino en sistemas operativos relativamente nuevos, tales como el MS-DOS

1969 Dennis Ritchie y Ken Thompson, de los Laboratorios Bell de Estados Unidos, desarrollan un OS a su propia conveniencia para operar en un ordenador PDP-7. Lo bautizan "Unix" haciendo un juego de palabras con el gran OS Multics en uso en aquel entonces

1971 Se transfiere a ordenadores PDP-11

1973 Las versiones originales estaban escritas en lenguaje ensamblador. Casi en su totalidad se recodificó en c, haciéndolo fácilmente portable a otros ordenadores, para lo que sólo se requiere un compilador de c

1974-75 Se introduce el PWB/Unix, que, en esencia, es una versión grande del Unix

1977 Se desarrolló la versión 7 para liberar al Unix de características dependientes de la máquina. Ésta es la versión que se utiliza actualmente

nadores como Xenix, Cromix, Onyx, Idris, Coherent y OS-9, todos los cuales tienen muchas características en común con el Unix. Ahora, sistemas operativos como el CP/M y el MS-DOS están adoptando características estilo Unix.

El Unix puro tal vez no llegue nunca a ser un éxito comercial, pero ha causado un impacto profundo en el desarrollo de sistemas operativos y seguirá siendo importante durante mucho tiempo. Las versiones actuales incluyen la versión 7 de Laboratorios Bell, que pronto se sustituirá por la versión 5 y el derivado desarrollado en la Universidad de California en Berkeley, que se ha extendido en la versión 4.2. Todos los ejemplos citados aquí están tomados de la versión Berkeley 4.2, pero se aplican sin dificultad a la mayoría de las demás versiones.

Una característica fundamental del Unix es el caparazón, la parte del OS con la que se comunica el usuario. Es mucho más que el simple procesador de comandos de muchos sistemas, porque tanto el proveedor del sistema como el usuario individual pueden confeccionarlo a medida para incorporar nuevas instrucciones y una interface para el usuario distinta si así se requiriera. También proporciona muchas de las facilidades de un lenguaje de programación como el c, y junto con la gran variedad de herramientas disponibles se pueden escribir programas largos y complejos sin utilizar un lenguaje de programación en absoluto.

Inicialmente veremos algunos de los comandos estándares y luego veremos las formas en que se crean comandos y nombres nuevos para comandos ya existentes. El formato de la mayoría de los comandos Unix es el mismo:

```
nombre_instrucción opciones archivo_o_
nombre_directorio
```

donde cada opción empieza con un signo menos. Un hecho importante a recordar sobre el Unix es que es sensible a los tipos de letra, es decir, los nombres "JUAN", "Juan" y "juan" se considerarán todos diferentes. Es un error común utilizar letras mayúsculas para un comando y encontrarse con que éste no funciona (la mayoría de los comandos Unix estándares utilizan letras en minúsculas).

La primera etapa en la utilización de un sistema Unix (o, para el caso, de cualquier otro sistema multiusuario), es *conectarse* (hacer *log in*). Esto se realiza automáticamente siempre que se conecta el sistema, pero a partir de entonces se puede hacer en cualquier momento digitando la instrucción login. El sistema le proporcionará el siguiente aviso:

```
login:
```

En cuyo momento usted debe digitar su nombre de identificación de usuario exclusivo. El sistema responde con:

```
password:
```

en cuyo punto usted digita su propia contraseña, que no se refleja en la pantalla. Las contraseñas normalmente tienen al menos seis caracteres de longitud. Si comete un error al entrar su contraseña, la debe repetir, pero para impedir que personas no autorizadas descubran contraseñas por el método de ensayo y error el sistema podría negarse a dejarlo probar otra vez al cabo de un cierto número de errores. Una vez completado el procedimiento, por lo general el sistema emitirá un mensaje de bienvenida y, finalmente, el aviso del OS.

En esta etapa el Unix ejecutará el propio programa de caparazón hecho a medida por usted y contenido en un archivo especial, .login, que entre otras cosas podría definir su propio aviso personal. El aviso Unix estándar es %. Ahora a usted se lo reconocerá como un usuario autorizado y el Unix le habrá conectado automáticamente a su propia área del disco donde se conservan todos sus archivos. Es posible acceder a archivos de otras áreas, en particular aquellos designados como públicos, pero, si se requiriera, usted podría proteger archivos y áreas determinadas para impedir el acceso no autorizado. Si desea cambiar su contraseña, lo que es aconsejable de vez en cuando, podría hacerlo digitando la instrucción passwd, que le solicitará que digite primero su contraseña antigua y después su nueva contraseña dos veces.

La mejor forma de conocer a un nuevo sistema de ordenador es probándolo y el Unix proporciona dos útiles facilidades para ayudarle a conocer el sistema. El Unix es un caso casi único en sistemas operativos por el hecho de que incluye un manual en línea con entradas para virtualmente todas las instrucciones o temas que usted pudiera querer utilizar. La instrucción que permite al usuario ver una entrada del manual es:

```
man nombre_tema
```

Para los usuarios novatos, el Unix incorpora un programa de autoformación que se inicializa mediante la instrucción learn, que ofrece lecciones sobre diversos temas.

Para dar por terminada una sesión con el Unix, se debe impartir la instrucción logout; ésta le desconectará del sistema hasta su próximo login.

Juega la banca

Nos dedicaremos a considerar las rutinas que permiten que el ordenador replique inteligentemente al jugador

Después de que el jugador ha completado su mano, le llega su turno a la banca. Ésta disfruta de varias ventajas en la versión de las reglas del veintiuno que hemos adoptado para nuestro programa. En primer lugar, la banca conoce los marcadores que han obtenido los jugadores y, por tanto, no tiene que aventurar peticiones de cartas arriesgadas que pudieran hacer que la mano se pasara. Segunda ventaja: sólo necesita empatar al jugador para ganar la vuelta y quedarse con su dinero. Para atenuar un poco estas ventajas, el programa no permite que la banca queme una mano de 12, 13 o 14.

En esta etapa del juego la pantalla visualizará los naipes del jugador y los dos naipes que se repartieron inicialmente a la banca. El primero de éstos se repartió cara abajo, de modo que la siguiente tarea del programa es dar vuelta el naipe. La forma más simple de hacerlo es borrar los naipes de la banca, utilizando la rutina de borrado general de la línea 670, y después reimprimirlos boca arriba estableciendo CN y SU (los números de naipe y palo) en los valores retenidos en la sección de la banca de la matriz de manos, HD(,,), y por último llamar a la rutina de visualización de naipes que desarrollamos anteriormente en esta serie. Todo esto se lleva a cabo entre las líneas 150 y 170 del bucle principal del programa. Ahora el programa está en condiciones de jugar automáticamente la mano de la banca.

Si el jugador ha pasado, no es preciso emprender

El turno del ordenador

Gama Amstrad CPC

```
130 REM **** turno del ordenador ****
140 pl=2
150 ep=20:GOSUB 670:REM borrar naipes
160 cn=hd(pl,1,1):su=hd(pl,1,2):GOSUB 1000:REM
    reimprimir
170 cn=hd(pl,2,1):su=hd(pl,2,2):GOSUB 1000:REM
    reimprimir
180 GOSUB 2500:REM la banca pide etc
190 REM **** ganar o perder ****
200 IF bw=1 THEN GOSUB 700:PEN negro:PRINT"lo siento
    pero pierdes":GOTO 300
210 GOSUB 700:PEN negro:PRINT"ganas tu"
300 a$="":WHILE a$="";a$=INKEY$:WEND
305 REM ** si shift/s entonces barajar **
310 IF a$="S" THEN GOSUB 700:PRINT"barajando...por favor
    espera":GOSUB 3000
320 GOTO 50
2500 REM **** turno de la banca ****
2520 ON pv+1 GOSUB 2540, 2550, 2560, 2570, 2580
2530 RETURN
2540 bw=1:RETURN:REM apostador se pasa
2550 ts=ps:GOSUB 5000:RETURN:REM pedir hasta vencer
     apostador o pasarse
2560 ts=21:GOSUB 5000:RETURN:REM pedir hasta pontoon o
     pasarse
2570 GOSUB 5200:RETURN:REM pedir hasta juego de 5 naipes
     o pasarse
2580 GOSUB 800:IF ef=2 THEN bw=1:RETURN:REM royal
     pontoon de la banca
2585 bw=0:RETURN:REM royal pontoon del apostador
5000 REM **** la banca pide hasta alcanzar objetivo o pasarse
     ****
5010 GOSUB 800:REM evaluar
5020 IF ef=4 THEN bw=0:RETURN:REM se pasa
5025 IF ef=2 OR ef=5 THEN bw=1:RETURN:REM royal
     pontoon/juego de cinco naipes de la banca
5040 IF tt(2,1)> =ts OR (tt(2,2)<=21 AND tt(2,2)> =ts)
     THEN bw=1:RETURN
5045 IF hp(2)>5 THEN bw=0:RETURN:REM repartidos cinco
     naipes
5050 GOSUB 1300:GOTO 5000:REM repartir y volver a
     evaluar
5200 REM **** pedir hasta juego de 5 naipes o pasarse ****
5220 GOSUB 800:REM evaluar
5225 IF ef=4 THEN bw=0:RETURN:REM se pasa
5227 IF ef=2 OR ef=5 THEN bw=1:RETURN:REM royal
     pontoon/juego de cinco naipes de la banca
5228 IF hp(2)> 5 THEN bw=0:RETURN:REM repartidos cinco
     naipes
5230 GOSUB 1300:GOTO 5200:REM repartir naipe y volver a
     evaluar
```

Sinclair Spectrum

```
130 >REM **** TURNO DEL ORDENADOR ****
140 LET PL=2
150 LET EP=14: GO SUB 670: REM BORRAR NAIPES
160 LET CN=D(PL,1,1): LET SU=D(PL,1,2): GO SUB 100:
    REM REIMPRIMIR
170 LET CN=D(PL,2,1): LET SU=D(PL,2,2):GO SUB 1000:
    REM REIMPRIMIR
180 GO SUB 2500: REM LA BANCA PIDE ETC
190 REM **** GANAR O PERDER ****
200 IF BW=1 THEN GO SUB 700: PRINT "LO SIENTO PIERDES
    TU": GO TO 300
210 GO SUB 700: PRINT " GANAS TU# ";BT
300 LET A$=INKEY$: IF A$="" THEN GO TO 300
305 REM ** SI SYM/S ENTONCES BARAJAR **
310 IF A$=CHR$ 195 THEN GO SUB 700: PRINT
    "BARAJANDO... ESPERA POR FAVOR": GO SUB
    3000
320 GO TO 50
2500 > REM **** TURNO DE LA BANCA ****
2520 GO SUB (PV*10)+ 2540
2530 RETURN
2540 LET BW=1: RETURN: REM APOSTADOR SE PASA
2550 LET TS=PS: GO SUB 5000: RETURN: REM PEDIR HASTA
     VENCER AL APOSTADOR O PASARSE
2560 LET TS=21: GO SUB 5000: RETURN: REM PEDIR HASTA
     PONTOON O PASARSE
2570 GO SUB 5200: RETURN: REM PEDIR HASTA JUEGO DE
     CINCO NAIPES O PASARSE
2580 GO SUB 800/ IF EF=2 THEN LET BW=1: RETURN: REM
     ROYAL PONTOON
2585 LET BW=0: RETURN: REM ROYAL PONTOON DEL
     APOSTADOR
5000 >REM **** LA BANCA PIDE HASTA ALCANZAR
     OBJETIVO O PASARSE
5010 GO SUB 800: REM EVALUAR
5020 IF EF=4 THEN LET BW=0: RETURN: REM SE PASA
5025 IF EF=2 OR EF=5 THEN LET BW=1: RETURN
5040 IF T(2,1)>=TS OR (T(2,2)<=21 AND T(2,2)>=TS)
     THEN LET BW=1: RETURN
5050 GO SUB 1300: GO TO 5000: REM REPARTIR Y VOLVER A
     EVALUAR
5200 REM **** PEDIR HASTA JUEGO DE 5 NAIPES O PASARSE
     ****
5220 GO SUB 800: REM EVALUAR
5225 IF EF=4 THEN LET BW=0: RETURN: REM SE PASA
5227 IF EF=2 OR EF=5 THEN LET BW=1: RETURN
5228 IF P(2)> 5 THEN LET BW=0: RETURN
5230 GO SUB 1300: GO TO 5200
```

ninguna acción. La banca ha ganado y, obviamente, no hay necesidad alguna de naipes extras.

Si el jugador posee un marcador de menos de 21, la banca debe pedir repetidamente hasta igualar o vencer el marcador o hasta pasarse.

Si el jugador tiene *pontoon*, la banca debe pedir hasta obtener 21 o pasarse.

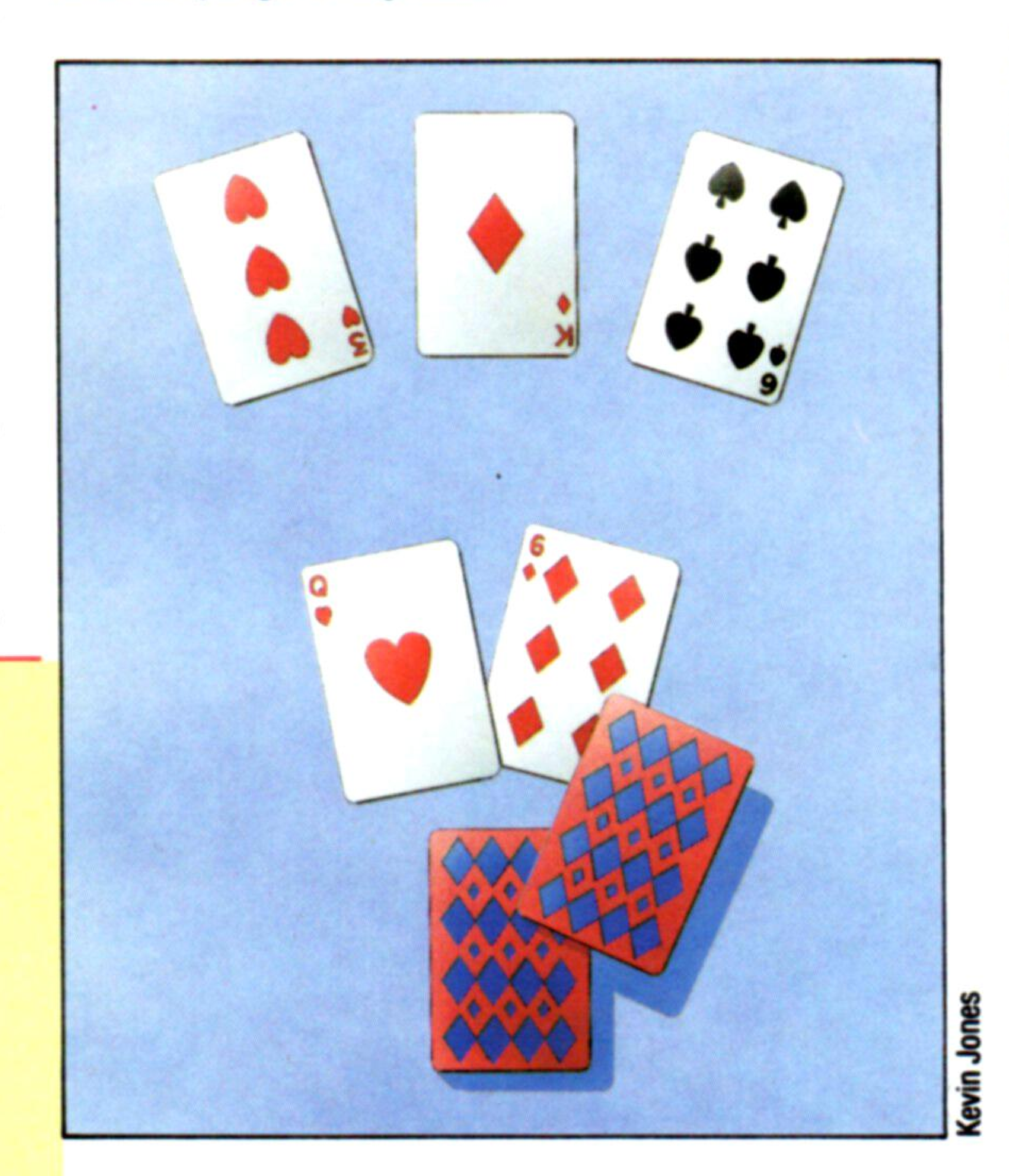
Si el jugador posee un juego de cinco naipes, debido a que el juego de cinco naipes vence al *pontoon* común, la banca debe pedir en un intento por obtener los cinco naipes, pero, por supuesto, puede pasarse en el intento.

Si el jugador tiene *royal pontoon*, la banca sólo puede ganar obteniendo un *royal pontoon* con las dos cartas ya repartidas.

Todas estas acciones, por supuesto, son innecesarias si la banca tiene *royal pontoon*, estado que no se puede superar.

La subrutina de la línea 2500 selecciona el curso de acción, utilizando una instrucción ON...GOSUB junto con la variable PV. En el caso de pedir para superar el marcador del jugador, se establece un marcador meta TS y se llama a otra subrutina, en la línea 5000. Si la banca ha de pedir para obtener un juego de cinco naipes, se necesita una rutina ligeramente diferente y ésta se halla en la línea 5200. En todos los casos el turno de la banca terminará cuando ésta gane o pierda, lo que se indica mediante el establecimiento en 1 o 0, según el caso, de una bandera BW.

El valor de esta bandera determina el mensaje de final del juego a imprimir.



Estado de cuenta de la banca
En esta versión del veintiuno o *pontoon*, la banca disfruta de la ventaja de saber cuál es la mano que debe superar. En este juego de ejemplo, el apostador se ha plantado en 19. La banca sabe que se debe repartir a sí misma al menos un naipe más en un intento por ganar la partida

Commodore 64

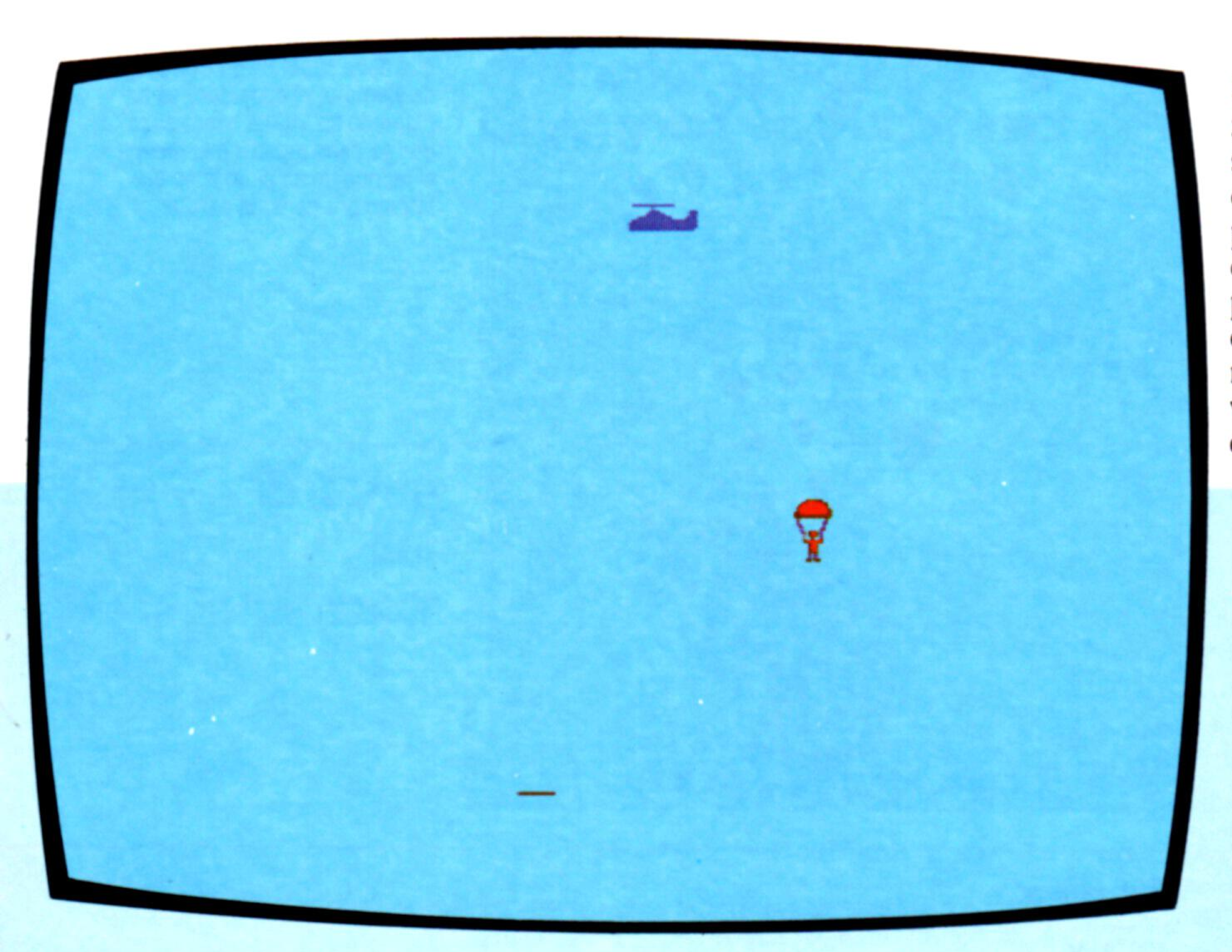
```
130 REM **** TURNO DEL ORDENADOR ****
140 PL=2
150 EP=20:GOSUB 670:REM BORRAR NAIPES
160 CN=HD(PL,1,1):SU=HD(PL,1,2):GOSUB 1000:REM
    REIMPRIMIR
170 CN=HD(PL,2,1):SU=HD(PL,2,2):GOSUB 1000:REM
    REIMPRIMIR
180 GOSUB 2500:REM LA BANCA PIDE ETC
190 REM **** GANAR O PERDER ****
200 IF BW=1 THEN GOSUB 700:PRINT CHR$(156);"LO
    SIENTO TU PIERDES":GOTO 300
210 GOSUB 700:PRINT CHR$(156);"GANAS TU"
300 GET A$:IF A$="" THEN 300
305 REM ** SI SHIFT/S ENTONCES BARAJAR **
310 IF A$=CHR$(211) THEN GOSUB 700:PRINT"BARAJANDO...
    ESPERA POR FAVOR": GOSUB 3000
320 GOTO 50
2500 REM **** TURNO DE LA BANCA ****
2520 ON PV+1 GOSUB 2540, 2550, 2560, 2570, 2580
2530 RETURN
2540 BW=1:RETURN:REM EL APOSTADOR SE PASA
2550 TS=PS:GOSUB 5000:RETURN:REM PEDIR HASTA
     SUPERAR AL APOSTADOR O PASARSE
2560 TS=21:GOSUB 5000:RETURN:REM PEDIR HASTA
     PONTOON O PASARSE
2570 GOSUB 5200:RETURN:REM PEDIR HASTA JUEGO DE 5
     NAIPES O PASARSE
2580 GOSUB 800:IF EF=2 THEN BW=1:RETURN:REM RUYAL
     PONTOON
2585 BW=0:RETURN:REM ROYAL PONTOON DEL
     APOSTADOR
5000 REM **** LA BANCA PIDE HASTA ALCANZAR OBJETIVO
     O PASARSE ****
5010 GOSUB 800:REM EVALUAR
5020 IF EF=4 THEN BW=0: RETURN:REM SE PASA
5025 IF EF=2 OR EF=5 THEN BW=1:RETURN
5040 IF TT(2,1)> =TS OR(TT(2,2)<=21 AND TT(2,2)>=TS)
     THEN BW=1:RETURN
5045 IF HP(2)> 5 THEN BW=0:RETURN:REM REPARTIDOS
     CINCO NAIPES
5050 GOSUB 1300:GOTO 5000:REM REPARTIR Y VOLVER A
     EVALUAR
5200 REM **** PEDIR HASTA 5 NAIPES O PASARSE ****
5220 GOSUB 800:REM EVALUAR
5225 IF EF=4 THEN BW=0:RETURN:REM SE PASA
5227 IF EF=2 OR EF=5 THEN BW=1:RETURN
5228 IF HP(2)> 5 THEN BW=0:RETURN
5230 GOSUB 1300:GOTO 5200:REM REPARTIR
     NAIPE
```

BBC Micro

```
130 REM
140 PL=2
150 EP=20:GOSUB 670
160 CN=HD(PL,1,1):SU=HD(PL,1,2):GOSUB 1000
170 CN=HD(PL,2,1):SU=HD(PL,2,2):GOSUB 1000
180 GOSUB 2500
190 REM
200 IF BW=1 THEN GOSUB 700:PRINT"LO SIENTO, PIERDES
    TU!":GOTO 300
210 GOSUB 700:PRINT"GANAS TU "
300 A$=GET$
305 REM
310 IF A$="S" THEN GOSUB 700:PRINT"BARAJANDO...
    ESPERA POR FAVOR":GOSUB 3000
320 GOTO 50
2500 REM
2520 ON PV+1 GOSUB 2540, 2550, 2560, 2570, 2580
2530 RETURN
2540 BW=1:RETURN
2550 TS=PS:GOSUB 5000:RETURN
2560 TS=21:GOSUB 5000:RETURN
2570 GOSUB 5200:RETURN
2580 GOSUB 800:IF EF=2 THEN BW=1
2585 BW=0:RETURN
5000 FOR DL=1 TO 100:NEXT
5010 GOSUB 800
5020 IF EF=4 THEN BW=0:RETURN
5025 IF EF=2 THEN BW=1:RETURN
5040 IF TT(2,1)> =TS OR (TT(2,2)<=21 AND TT(2,2)>=TS)
     THEN BW=1:RETURN
5050 GOSUB 1300:GOTO 5000
5200 REM
5210 GOSUB 1300
5220 GOSUB 800
5225 IF EF=4 THEN BW=0
5230 IF EF<>5 THEN 5210
5240 BW=1:RETURN
```

Paracaídas

Se han escrito versiones de este juego de acción para numerosos modelos de microordenadores. En esta ocasión presentamos el listado destinado al M05 de Thomson



Saltando de un helicóptero en vuelo, intente alcanzar el blanco situado en el suelo. Una primera pulsación sobre una tecla le permite descender verticalmente en caída libre. Una segunda pulsación produce la abertura del paracaídas. El descenso continúa más lentamente y en un ángulo de 45°, puesto que el viento le empuja. Cuanto mayor sea el tiempo que espere para abrir el paracaídas, tanto menor será la desviación. Pero no aguarde excesivamente, ya que, por debajo de 100 m, el paracaídas no se abrirá...

```
10 REM ***************
20 REM * PARACAIDAS *
30 REM ***************
40 GOSUB 470
50 HH=HH-4
60 IF HH=0 THEN PUT (0,1)-(27,9),R
70 IF HH=0 THEN HH=288
80 PUT (HH,1)-(HH+27,9),H
90 D$=INKEY$
100 IF D$="" THEN 140
110 IF PV>100 THEN 140
120 IF SP=1 THEN OP=1 ELSE SP=1
130 IF OP=0 THEN PV=10:PH=HH
140 IF SP=0 THEN 230
150 IF OP=0 THEN PV=PV+8
160 IF OP=1 THEN PV=PV+1:PH=PH-1
170 IF PV>167 OR PH<1 THEN 260
180 IF OP=1 THEN 210
190 PUT (PH,PV)-(PH+14,PV+23),PF
200 GOTO 50
210 PUT (PH,PV)-(PH+14,PV+23),PO
220 GOTO 50
230 FOR I=1 TO 50
240 NEXT I
250 GOTO 50
260 IF ABS(PH-A)>4 THEN 320
270 FOR I=1 TO 1000
280 NEXT I
290 S=S+10
300 GOSUB 670
310 GOTO 50
320 CLS
330 SCREEN 1,2,2
340 LOCATE 10,10
350 ATTRB 1,1
360 PRINT "SCORE :";S;
370 LOCATE 10,16
380 PRINT "OTRA ?";
390 ATTRB 0,0
400 IF INKEY$<>"" THEN 400
410 D$=INKEY$
420 IF D$="" THEN 410
430 IF D$<>"N" THEN RUN
440 CLS
450 SCREEN 4,6,6
460 END
470 LOCATE 0,0,0
480 SCREEN 4,6,6
490 DEFINT A-Z
500 DIM H(27,8)
510 DIM PF(14,23)
520 DIM PO(14,23)
530 DIM R(27,8)
540 CLS
550 DRAW "C4BM51,50R14L6G4L2G1L1G1D1F1R2
    1E1U5L2D1G2L5H1L1H1L1H1L1"
560 PAINT (58,56),4
570 GET (50,50)-(77,58),H
580 GET (100,50)-(127,58),R
590 CLS
600 DRAW "C1BM54,50G1L1G1D1G1D1R12U1H1U1H1L1H1L4"
610 PAINT (56,54),1
620 DRAW "C1BM51,56D1F1D1F1D3R2D6L1R1U3R
    2D3R1L1U4L1U4L1U1R2D1L1D4R1U2R2U3E1U1E1U1"
630 GET (50,46)-(64,69),PO
640 CLS
650 DRAW "C1BM54,61R1D6L2D2U2R3D2L1D1R2U
    1L1U2R3D2U2L3U3D3R1U6R1"
660 GET (50,46)-(64,69),PF
670 CLS
680 HH=288
690 HV=1
700 A=INT(RND*191)+10
710 DRAW "C0BM"+STR$(A)+",191R12"
720 SP=0
730 OP=0
740 PV=0
750 PH=0
760 RETURN
```

El Amstrad PCW 8256, conocido familiarmente como "Joyce", constituye un completo sistema de tratamiento de textos



Una cita con Joyce

Amstrad Consumer Electronics tiene la tradición de empaquetar tecnología ya existente en sistemas "todo en uno". El Amstrad PCW 8256, si bien es una máquina muy distinta a sus predecesoras, se ha construido en gran parte siguiendo la misma línea. Diseñado como paquete integrado para tratamiento de textos, el sistema incluye en su precio una unidad de disco incorporada, una impresora matricial y el software para tratamiento de textos.

Aun teniendo en cuenta la filosofía de fabricación y marketing de la empresa, el lanzamiento del PCW 8256, apodado "Joyce", nombre de la secretaria de Alan Sugar, causó sorpresa en la industria. Mientras que los ordenadores Amstrad anteriores estaban diseñados para atraer al mercado más amplio posible, la última máquina es la primera de la empresa dirigida a un sector específico, si bien más grande. De modo, entonces, que aunque el Amstrad CPW 8256 se puede utilizar como ordenador estándar, la empresa lo está dirigiendo básicamente a las pequeñas empresas.

Al igual que anteriores ordenadores Amstrad, la pantalla está alojada en una carcasa con la fuente de alimentación al costado de una unidad de disco incorporada y un conector que le proporciona potencia a la impresora independiente. Un cable enrollado que sale del teclado se enchufa en un conector lateral.

Las 82 teclas incluyen el grupo QWERTY estándar más algunas adicionales, tales como Alt y Extra, para proporcionar juegos de caracteres alternativos. Éstos incluyen caracteres griegos y otros, tales como ü, que permiten que el ordenador procese toda una variedad de lenguas. Inmediatamente a la derecha de éstas hay cuatro teclas de función programables, que poseen numerosos usos y proporcionan hasta ocho funciones diferentes. Hay un grupo de teclas de cursor en la esquina inferior derecha del teclado y arriba hay varias teclas dedicadas a diversas funciones de tratamiento de textos. Si bien representa una mejora respecto al teclado estándar de Amstrad, las teclas resuenan un poco cuando se entra texto a cualquier velocidad.

Pantalla verde

La pantalla es un modelo de fósforo verde y, según Amstrad, será el único tipo disponible. Con una resolución de 92 por 32 caracteres, la pantalla es más grande que la que se proporciona en la mayoría de los ordenadores. En pantalla, los caracteres son del mismo tipo de letra, pero más grandes que los de otros modelos Amstrad, con lo que el texto resulta más fácil de leer. La unidad de disco se halla en la esquina superior derecha de la carcasa de la pantalla, debajo de la cual hay otra ranura para instalar una segunda unidad de disco, detrás de la placa donde está inscrita la marca.

La impresora matricial de cinco por ocho agujas que se suministra junto con el PCW 8256 es capaz de dar cabida a hojas A4 individuales o al formulario continuo estándar de 11 pulgadas. La primera de las dos modalidades en las que puede operar se conoce como "modalidad de borrador" y, tal como sugiere su nombre, es ideal para producir copias en borrador, memorándums sencillos, etc. Esta modalidad, no obstante, ilustra claramente las limitaciones de la impresora. Una observación atenta revela que los puntos individuales son incapaces de formar una línea continua.

Amstrad no fabrica su propia impresora, sino que "emula modelos" de otras empresas, política que ha suscitado algunas quejas de parte de los usuarios. Lo que es más importante, sin embargo, es que una impresora de buena calidad es esencial para un sistema de tratamiento de textos, de modo que sería lícito cuestionar la decisión de Amstrad de utilizar este modelo en particular con el PCW 8256. Aparentemente la empresa ha reconocido esta limitación y ha anunciado que proveerá una interface RS232 y Centronics que permitirá la instalación en la máquina básica de impresoras de más alta calidad.

Usted, sin embargo, se enfrentará con varios problemas si decide instalar otra impresora. Debido a que *LocoScript*, el software para tratamiento de textos empaquetado con la máquina, se basa en gran medida en el hardware existente, posee dificultades para comunicarse con otras impresoras. La solución de Amstrad es modificar el sistema opera-

Amstrad amplía su horizonte
El PCW 8256, apodado "Joyce", es el primer micro de Amstrad pensado principalmente para el mercado de gestión. Si bien está siendo comercializado como procesador de textos exclusivo (con pantalla, impresora, unidad de disco, y software para tratamiento de textos empaquetado), el PCW 8256 también es un ordenador de ocho bits muy potente

tivo CP/M 3.0 para poder leer los archivos en formato ASCII y enviarlos a la impresora. Esto significa, por supuesto, que usted tendrá que salir del *LocoScript* para poder imprimir el archivo. Además, muchas de las facilidades para formateo de la impresión más atractivas del *LocoScript* no se pueden incluir en tal sistema.

Sin embargo, la modalidad de impresión de "alta calidad" supera estas limitaciones de hardware. La impresora para dos veces por cada línea, rellenando los agujeros dejados por los puntos, de modo muy similar al sistema usado para imprimir en negrita. Aunque esto mejora enormemente la calidad de la impresión, reduce de manera drástica la velocidad de la impresora, que pasa de 90 caracteres por segundo (cps) en modalidad de borrador, a 20 cps. Existen algunas dudas respecto a la fiabilidad de la impresora. La fila central de agujas del sistema que revisamos falló tras apenas dos días de utilización.

Al igual que los anteriores ordenadores de Amstrad, el PCW 8256 se basa en el conocido procesador Z80 equipado con 256 Kbytes de RAM. La máquina incorpora las ahora comunes técnicas de *conmutación de bancos*, ya que de lo contrario sería incapaz de manipular tanta memoria. Se han organizado alrededor de 110 Kbytes como un disco de silicio, que el procesador trata como un disco flexible y, por tanto, accede a la información consiguientemente. Dado que la información está retenida en chips de RAM, se puede acceder a la misma casi instantáneamente, lo que, por supuesto, es más rápido que desde un disco.

El software empaquetado

El software empaquetado con el PCW 8256 incluye CP/M 3.0, una versión mejorada del OS de anteriores máquinas Amstrad. También se incluye el sistema operativo de disco AMSDOS de Amstrad, junto con el paquete para gráficos GSX de Digital Research. Éste permite que el CP/M visualice gráficos (facilidad que originalmente no se había incorporado en su diseño) y ofrece la posibilidad de que varios paquetes de gestión disponibles bajo CP/M hagan uso de la facilidad de gráficos.

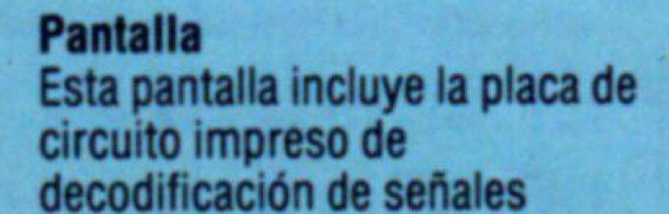
Asimismo, Amstrad ha empaquetado con la máquina el Mallard BASIC, el Dr LOGO y el *LocoScript*. Amstrad afirma que el *LocoScript* se ha diseñado para que sea potente y a la vez tan fácil de usar como una máquina de escribir eléctrica.

Tras el encendido, la pantalla *LocoScript* se llena de marcas de TAB, pensadas para ayudar a formatear el texto. No obstante, a muchos usuarios les parecerá que esto produce una visualización más bien confusa, que se interpone en medio del texto. Pero los programadores han permitido que el usuario elimine todas estas marcas, con lo que se obtiene una visualización más clara.

El sistema, por cierto, parece ser tan potente y flexible como muchos sistemas para tratamiento de textos diseñados para máquinas de gestión de gran calidad. *LocoScript* posee facilidades que permiten desplazar el texto dentro de un documento, trasladar el cursor por caracteres, palabras o párrafos, y localizar palabras y frases, tareas todas estas que se llevan a cabo mediante las teclas especializadas en tratamiento de textos incorporadas en el teclado. Para copiar un área de texto de una parte de un documento en otra parte, por ejemplo, es necesario desplazar el cursor hasta el comienzo del área en cuestión y pulsar la tecla COPY. El proceso se repite luego al final de esa sección. La transferencia se efectúa desplazando el cursor hasta la posición correspondiente y pulsando la tecla Paste.

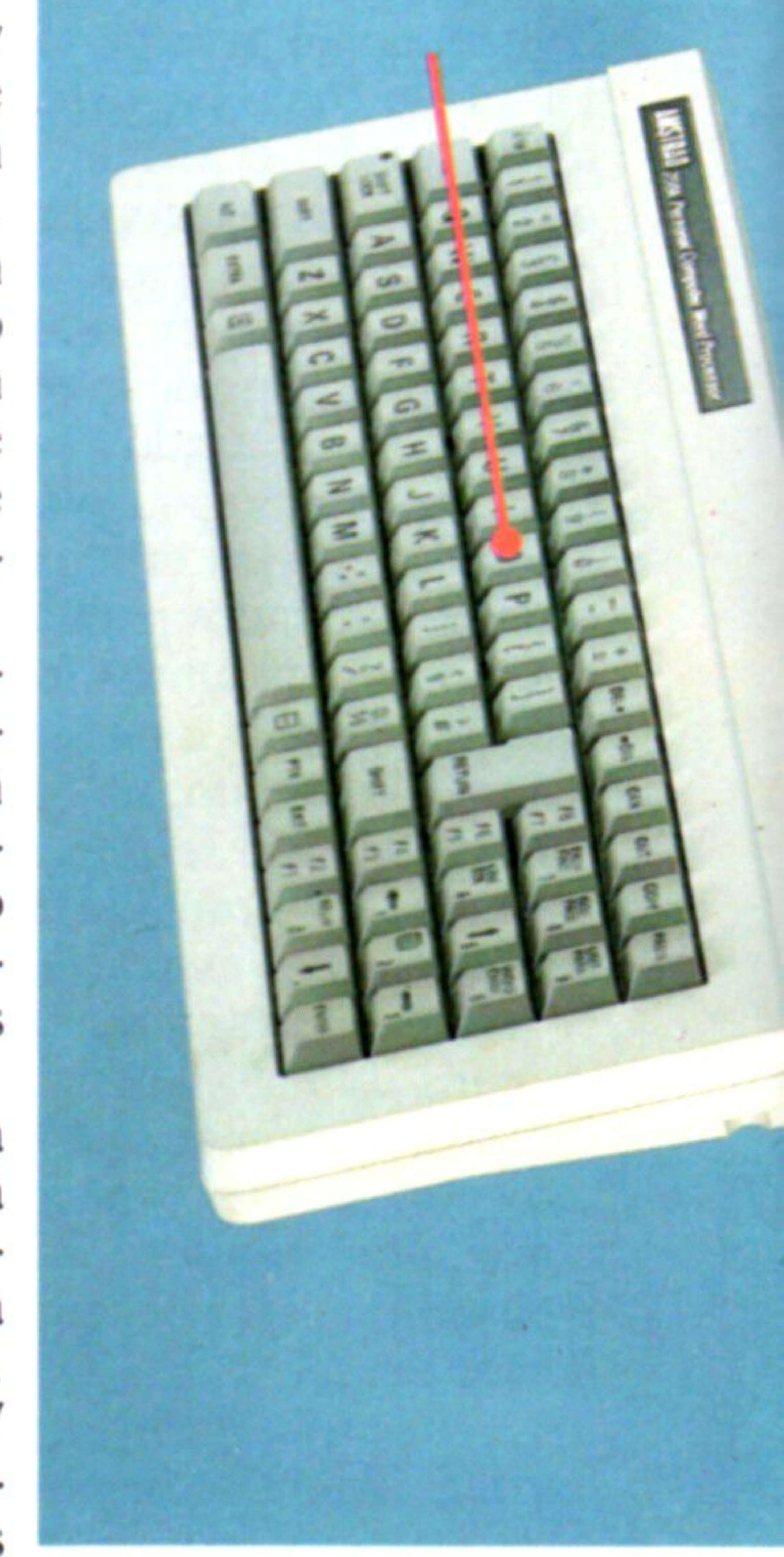
A las facilidades tales como alineación, enfatización de caracteres y formatos de impresión se accede a través de las teclas de función. Aparecerá un menú con una serie de opciones que se pueden seleccionar utilizando las teclas del cursor y pulsando luego la tecla Enter. Este sistema de menús y submenús se ha diseñado para aumentar las facilidades para tratamiento de textos.

Como hemos visto en nuestra serie dedicada a los paquetes para tratamiento de textos, la mayoría de éstos hacen un uso intensivo de caracteres Control y menús de pantalla separados para llevar a cabo la amplia variedad de funciones disponibles. Es comprensible que los programadores del PCW 8256 hayan optado por utilizar este sistema de teclas de función programables, menús y ventanas



Pantalla
Esta pantalla incluye la placa de circuito impreso de decodificación de señales

Teclado
Al igual que muchos otros procesadores de textos especializados, el PCW 8256 contiene algunas teclas adicionales inexistentes en ordenadores estándares, que se utilizan en tratamiento de textos





Facilidades en pantalla
El *LocoScript*, el software para tratamiento de textos diseñado a medida para el PCW 8256, soporta una amplia gama de facilidades. Si bien muchas de ellas están disponibles a través de una serie de ventanas situadas en la parte superior de la pantalla, muchas otras están incorporadas en el texto. Aunque es útil examinar estas instrucciones en pantalla, a muchos usuarios les parecerá que las mismas confunden el texto que se está escribiendo. Sin embargo, se las puede eliminar de la visualización en pantalla



ULA
Este chip fabricado a medida maneja todas las rutinas de control, incluyendo la visualización VDU y la gestión de memoria

Controlador de disco flexible
Este chip maneja las rutinas relacionadas con el control de archivos y el acceso desde la unidad de disco

Controlador de impresora
Este chip controla la impresora exclusiva

CPU
Al igual que otros ordenadores Amstrad, el PCW 8256 utiliza el procesador Z80A como su CPU

Unidad de disco
El PCW 8256 tiene instalada como estándar una única unidad de disco de 3 pulgadas

Chips de RAM
Originalmente la máquina fue diseñada para utilizar chips de memoria de 128 Kbits. Sin embargo, cuando aparecieron en el mercado chips económicos de 256 Kbits, se emplearon éstos, dejando vacíos la mitad de los conectores de RAM

PCB del ordenador
Amstrad ha podido ofrecer la máquina a un precio tan económico debido en parte al elegante diseño de la placa de circuito impreso del ordenador, que requiere una cantidad de chips notablemente baja

Chris Stevens

AMSTRAD PCW 8256

DIMENSIONES
375 × 326 × 309 mm

MEMORIA
RAM de 256 K, de los cuales 128 K están configurados como un disco de RAM

CPU
Z80

VISUALIZACION DE VIDEO
92 × 32 caracteres; 24 líneas × 90 caracteres, zona de texto en *LocoScript*

INTERFACES
Conector para segunda unidad de disco, interface para impresora, conector para teclado

SOFTWARE SUMINISTRADO
Versión 3.0 de CP/M, programa de tratamiento de textos *LocoScript*, Mallard BASIC, Dr LOGO, utilidad para gráficos GSX

DOCUMENTACION
El manual es exhaustivo, tal como nos tiene acostumbrados Amstrad con sus manuales

VENTAJAS
El PCW 8256 constituye una oferta muy interesante al incluir en su precio un ordenador de 256 K, pantalla, unidad de disco e impresora

DESVENTAJAS
Parte del hardware muestra signos de la reducción de costos. En especial, la impresora parece ser lenta y de dudosa fiabilidad. Además, parece ser que incluso con una interface especial una impresora diferente tampoco podría sacar partido de muchas de las facilidades de *LocoScript*

como un medio de simplificar el sistema. Aunque usted ya no tendrá que aprender los numerosos caracteres Control que utilizan la mayoría de los procesadores de textos, cada una de las teclas de función de la máquina Amstrad conduce a su propio menú, y algunas de las etiquetas tanto de las teclas como de los menús no dan una idea cabal de su finalidad. Esto significa que usted deberá aprender algunos pasos y pulsaciones de teclas para obtener el efecto deseado. (Observe que si bien el PCW 8256 no posee ninguna tecla Control, se sustituye la tecla Alt para gran número de programas basados en CP/M).

Entrada de comandos

LocoScript proporciona un método más rápido de entrar comandos. A los lados de la barra Space hay teclas marcadas ± y =. Para alinear una línea por la derecha, pulsando la tecla ± seguida por RJ se ejecutará la instrucción, mientras que = la dará por terminada. Esto, así y todo, lleva más tiempo que utilizar un único carácter de control; habría sido mucho más simple emplear un mecanismo *toggle*.

Se suma a la confusión parte de la edición en pantalla. Una vez que se ha modificado un párrafo, algunas de las líneas podrán parecer desiguales porque el software no habrá ajustado el párrafo, obligando al propio usuario a hacerlo pulsando la tecla Relay. Este proceso de ajuste o alineamiento manual es común en procesadores de textos más antiguos, en particular en el *WordStar*, pero cabría esperar que un paquete desarrollado en 1985 hubiera incorporado el ajuste automático.

Otra característica curiosa del *LocoScript* es que parecería que el OS del ordenador fuera incapaz de leer muchos de los archivos. Esto es consecuencia de la capacidad del CP/M 3.0 para soportar numerosos usuarios diferentes. Cuando se carga el OS, pasa automáticamente por defecto al usuario 0. Al buscar en los archivos listados bajo diferentes números de usuario, el operador se encontrará con los documentos *LocoScript* "desaparecidos".

A pesar de estos problemas, el PCW 8256 representa una buena oferta y contiene numerosas facilidades excelentes. La gama y la variedad de las instrucciones para la impresora y formateo de páginas son similares a las que hay disponibles en paquetes incluidos en micros que cuestan varias veces el precio de la máquina de Amstrad. Parece ser, sin embargo, que esta vez la política de Amstrad de ofrecer la máxima cantidad de facilidades por el mínimo precio se ha llevado a un límite extremo. Los usuarios quizá opinen que un diseño un poco mejor (en especial en el caso de la impresora), aun al costo de algunas de las utilidades, hubiera hecho del PCW 8256 una propuesta muchísimo más interesante.

Guardabosque

Los directorios jerárquicos representan un método muy práctico de manejar grandes cantidades de archivos

Todos los comandos residentes que analizamos en el capítulo anterior están disponibles en todas las versiones de MS-DOS. Sin embargo, desde la versión 2.0 en adelante se puede contar con una nueva gama de facilidades, la mayoría de las cuales han sido sacadas del Unix. Tuvo vital importancia la introducción de una estructura de *directorio jerárquico*. La misma resulta particularmente útil con los modernos discos flexibles de elevada densidad, y es absolutamente esencial para los sistemas de disco rígido.

Cuando se carga por primera vez un sistema DOS 2, el usuario "ve" una zona de la unidad de disco por defecto que contiene varios archivos. En la mayoría de los casos se trata de programas (.EXE o .COM) o bien archivos de datos. Es normal tener tanto como 720 Kbytes de almacenamiento en un microflexible moderno de 3½ pulgadas, y varias decenas de megabytes en un disco rígido. El gran número de archivos significa que un simple listado del directorio podría suponer la visualización de una enorme cantidad de información, la mayor parte de la cual es irrelevante para el campo de interés actual del usuario. Está claro que se requiere un método para descomponer la zona de disco en *subdirectorios* manejables.

El directorio por defecto se llama *raíz* y se puede crear un subdirectorio dentro de la raíz mediante la instrucción md (de mkdir, o M*a*K*e* DIR*ectory*: hacer directorio). De modo que:

```
md trabajo
```

crearía un "archivo" (denominado trabajo) en el directorio raíz que "contuviera" otros archivos de datos directos.

Quizá deseáramos crear algunos documentos o programas fuente dentro de este nuevo directorio, de modo que cambiamos el directorio (CH*ange* DIR*ectory*) impartiendo la instrucción chdir, o su alternativa:

```
cd work
```

Si ahora digitamos dir (para listar el contenido de nuestro "directorio actual"), podemos esperar que esté vacío. Sin embargo, el MS-DOS ya ha creado dos subdirectorios dentro de trabajo. Éstos se llaman . y .. (un tanto enigmáticamente) y se refieren, respectivamente, al directorio actual (trabajo) y su "padre". Por tanto, si entráramos el comando:

```
dir..
```

se ofrecería un listado de todos los archivos del directorio raíz. Ahora éste incluiría la entrada:

```
TRABAJO <DIR> 17-11-85   11.21 a
```

Si ahora damos los comandos:

```
mkdir hoy
chdir hoy
```

nos encontraríamos "en" un directorio que tendría como directorio padre a trabajo y a la raíz como su "abuelo".

Al especificar el nombre de cualquier archivo del sistema, debemos dar un *nombre de paso* completo si el archivo no está en el directorio actual. De modo que, por ejemplo, un programa del directorio raíz se podría especificar ya sea como ..\..\ prog o simplemente \prog (la barra invertida se utiliza como un separador en nombres de paso o, si es el primer símbolo de un nombre de paso, como una abreviatura para la raíz). El movimiento un nivel "hacia arriba" (hasta el subdirectorio trabajo) se puede conseguir tanto con chdir.. o, en este caso, con cd/trabajo.

Si ahora entráramos md fuente, conseguiríamos crear un directorio "hermano" de hoy, y ambos se considerarían como "hijos" de nuestro directorio actual, trabajo.

Como puede verse, esta estructura empieza a parecerse a un árbol genealógico y de hecho se denomina *árbol* de directorios. La adición de algunas ramas más podría producir la estructura en la siguiente página.

Por consiguiente, un nombre de paso completo para un archivo del directorio texto denominado MEMO.DOC podría ser:

```
trabajo\fuente\texto\memo.doc
```

Éste localizaría el archivo en cualquier parte del árbol, dado que se especifica un nombre de paso completo comenzando desde la raíz (\); no obstante, si actualmente estuviéramos en el subdirectorio hoy podríamos utilizar:

```
..\fuente\texto\memo. doc
```

En cualquier momento, un comando dir dará un listado del directorio actual, pero se puede especificar un camino alternativo, como en:

```
dir b:\pascal\fuente\prognuevos
```

Observe que en este caso el nombre final puede ser el de un directorio, no de un archivo, y que se podría dar una unidad de disco diferente como el primer elemento del camino (en este caso, la unidad b). Mientras no se quite un disco de una unidad, el disco "recordará" todo subdirectorio actualmente seleccionado en esa unidad.

Esto significa que:

```
copy\*.*b:
```

copiará todos los archivos del directorio raíz del disco actual en el directorio actual, cualquiera sea, de la unidad b. A cada disco o *volumen* se le puede dar un nombre, ya sea cuando se formatea originalmente el disco o mediante la utilidad transitoria LABEL.EXE.

LABEL.EXE y FORMAT.EXE son dos de los programas que se suministran como utilidades *transitorias*. Éstos son programas separados (por lo general llevan la ampliación .EXE, pero en algunas ocasiones .COM) que se pueden ejecutar entrando sus nombres primarios. Con el fin de formatear un

disco, por ejemplo, se puede ejecutar el programa FORMAT.EXE entrando:

```
format b:
```

La instrucción FORMAT toma dos "interruptores" opcionales delimitados por una barra. Las opciones son:

\s Para copiar el sistema MS-DOS tras el formateo.
\p Para preguntar el nombre de un volumen o "etiqueta" a dar al disco. Sólo se permite un archivo (raíz) que tenga este atributo.

Ello da por sentado que el directorio operativo actual contiene FORMAT.EXE; pero supongamos que no lo contenga. El DOS 2 proporciona un comando residente sumamente útil para posibilitar la definición de los nombres de camino por defecto:

```
path=A:\sistema;A:\sistema\utils
```

La entrada de un comando no reconocido hará que el sistema siga el camino (o serie de caminos alternativos separados mediante punto y coma) con el fin de encontrar el comando. Usted puede averiguar cuál es el camino de búsqueda actual entrando:

```
path
```

y si no se ha definido ninguno se imprimirá el mensaje No path.

Muchas versiones de MS-DOS a menudo se proporcionan con comandos externos adecuados para el sistema OEM, pero hay algunos tan útiles que se encuentran en todas las versiones. PRINT.EXE permite la impresión como tarea de fondo de un archivo de texto mientras el usuario continúa trabajando en el ordenador. EDLIN.EXE es un editor de líneas que permite la creación de archivos de texto sin necesidad de un procesador de textos adicional. DISKCOPY.COM (o DCOPY) copia el contenido completo de un disco, y DISKCOMP (o COMP) se puede utilizar para verificar una transferencia.

Todas las utilidades MS-DOS que suponen el uso conceptual de dos unidades de disco son suficientemente "inteligentes" como para comprender la necesidad del intercambio de discos cuando se ejecuta un entorno de disco único. Por ejemplo, DISKCOPY carga datos de un disco fuente en la memoria y luego imparte un aviso para cambiar discos. Una vez completa la operación de copiado, se da otro aviso para insertar el disco fuente; y así sucesivamente, hasta haber transferido el disco entero. Se detectan los sectores malos y el disco de destino se puede formatear a la vez que es copiado.

La utilidad más útil para comprobar discos es CHKDSK.EXE, que proporciona información completa sobre el estado de un disco, incluyendo la señalización de la existencia de "archivos ocultos". Empleado sin argumentos, el comando:

```
chkdsk
```

comprueba la unidad "conectada" actualmente y produce un informe con la siguiente forma:

```
720666 bytes espacio total de disco
 47528 bytes en dos archivos ocultos
  2048 bytes en cuatro directorios
526386 bytes en 39 archivos del usuario
144704 bytes disponibles en el disco

262144 bytes memoria total
198726 bytes libres
```

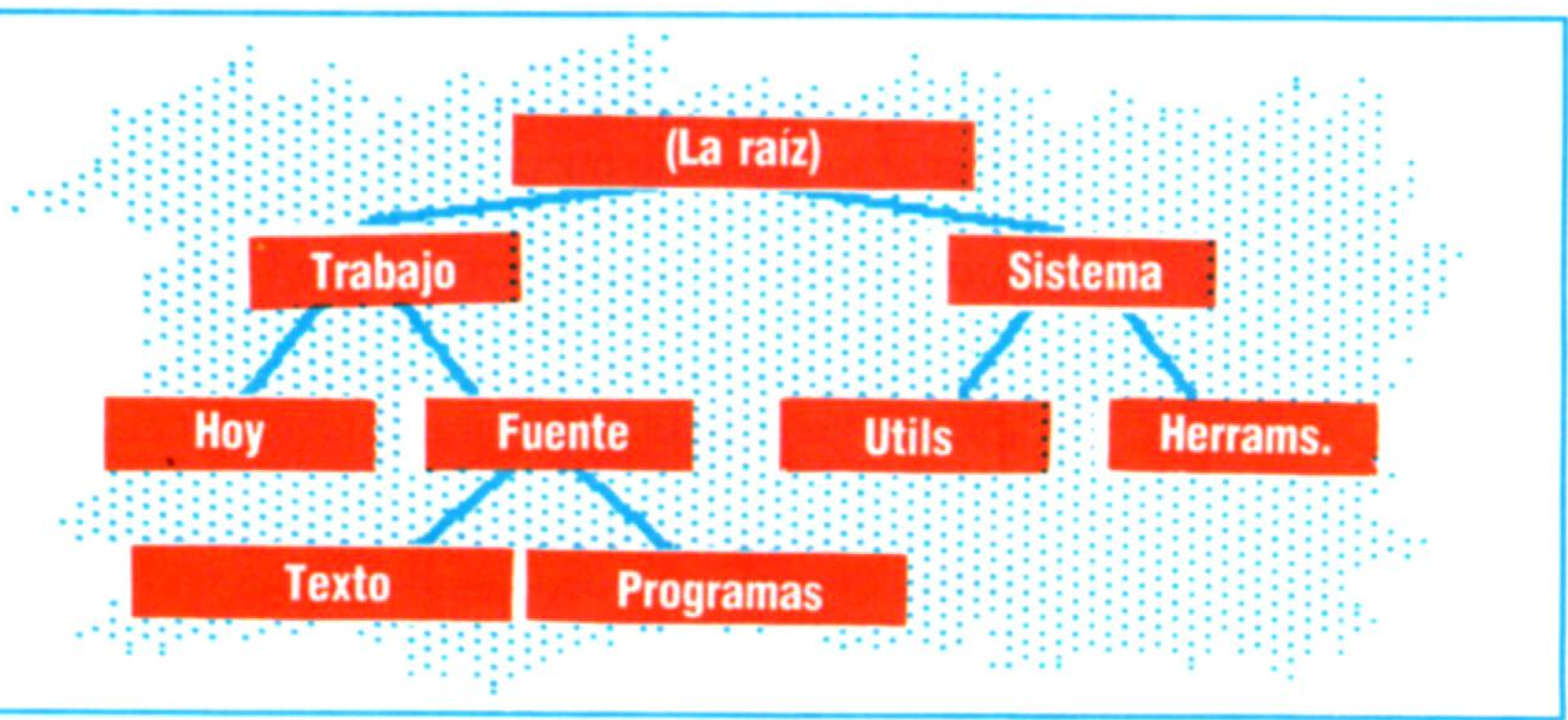
Es de gran utilidad que CHKDSK informe sobre el estado de la memoria del sistema (los dos últimos ítems) y que también compruebe el uso general del disco y la File Allocation Table, FAT (tabla de ubicación de archivos) retenida para cada dispositivo. El MS-DOS puede percibir cuándo se están cambiando los medios y reconstruir una FAT para un nuevo disco leyendo la información del sistema relativa al mismo.

Una palabra de advertencia: jamás use la utilidad CHKDSK del DOS suministrada con una versión de MS-DOS para comprobar discos DOS de otra versión importante. Los sectores del disco y la información sobre el directorio pueden estar dispuestos en un formato muy diferente, y esto puede ser especialmente peligroso si el DOS 1.x y el DOS 2.x están "vivos" en el mismo sitio. La mezcolanza resultante de enlaces entretejidos podría significar horas de trabajo infructuoso tratando de recuperar, pongamos por caso, un documento en el cual se hubieran intercalado "páginas" de código máquina de otro archivo. Si por descuido usted utiliza CHKDSK del DOS 1 en un sistema de disco rígido DOS 2 con 10 o 20 Mbytes en juego, las consecuencias pueden ser desastrosas.

Si usted tuviera motivos para mejorar un antiguo IBM PC o Sanyo pasando de DOS 1 a DOS 2, la moraleja es obvia: hacer copias de seguridad de todos los datos existentes en discos formateados no sistematizados, y después destruir todos los sistemas actuales (mediante reformateo si fuera necesario) antes de instalar DOS 2. Los OEM que suministran las mejores implementaciones de MS-DOS se aseguran de que las utilidades transitorias incorporen código para comprobar que el sistema en el que se estén ejecutando sean de una versión compatible, pero así y todo se debe ser cuidadoso.

División del trabajo

El sistema de directorio de archivos jerárquico implementado en el MS-DOS permite que los usuarios de sistemas de disco de gran capacidad dividan los archivos en subcategorías, dado que los directorios no sólo pueden contener nombres de archivos sino nombres de otros directorios. En este ejemplo el archivo "texto" es un miembro del directorio "fuente", que por su parte es miembro del directorio "trabajo". El directorio raíz contiene las principales categorías de grupos de archivos dentro de la estructura arborescente



Caroline Clayton

Conciencia de clase

El compilador del C reconoce a las variables como pertenecientes a una de cuatro diferentes clases

Además de ser de un tipo determinado, todas las variables de C poseen una *clase de almacenamiento*, que determina la forma en que el compilador las reconoce y asigna memoria para ellas. Hay cuatro clases: automática, externa, registro y estática. Las variables pueden llevar especificada su clase colocando las palabras clave auto, extern, register o static delante del tipo en la declaración.

Por ejemplo:

```
extern double x,y;
```

Casi todas las variables son *automáticas* por defecto. Toda variable declarada dentro del cuerpo de una función será automática por defecto y local de esa función. Por consiguiente, cada vez que se entre la función se asignará nuevo espacio de almacenamiento para esa variable y el mismo se perderá tras salir de la función. No existe modo alguno por el cual tal variable pueda conservar un valor entre dos llamadas a la función. Lo mismo se aplica a toda variable definida dentro de un bloque de código, indicado por estar encerrado entre llaves {}. Recuerde que main() es una función, de modo que incluso las variables declaradas aquí normalmente serán automáticas.

Las variables *externas*, por el contrario, son globales, es decir, se puede aludir a ellas en cualquier punto del programa, e incluso en ciertos casos a través de funciones que aparezcan en un archivo de código fuente distinto al de la declaración. Una variable declarada fuera del cuerpo de una función se considerará externa por defecto. Asimismo, una variable externa se puede declarar en cualquier otro punto dentro del cuerpo de una función o bloque. Continuará existiendo aún después de que acabe la ejecución de una función y se puede aludir a ella en cualquier punto del programa que sea posterior a la declaración.

Si se declaran variables externas del mismo nombre en dos o más archivos de código fuente, se considerarán como la misma variable cuando los archivos se unan entre sí. Si se declara una variable local con el mismo nombre que una variable global, "enmascarará" a la variable global dentro de su ámbito, de modo que las alusiones a ese nombre de variable serán a la variable local dentro de su ámbito y a la variable local fuera de él.

Las variables *de registro* se comportan exactamente como las variables automáticas, y de hecho en algunos casos se tratarán del mismo modo. Sin embargo, de ser posible, el compilador asignará para ellas espacio en almacenamiento de alta velocidad. Esto evidentemente dependerá mucho del procesador que se esté utilizando para ejecutar el programa, y en particular del tamaño y cantidad de sus registros. Un 68000, por ejemplo, con sus 16 registros de 32 bits para fines generales, posee alcance suficiente para utilizar los registros para una cantidad limitada de variables de registro, mientras que un Z80 probablemente no tendrá espacio. Las variables de registro sólo deben utilizarse de cuando en cuando y declararse lo más tarde posible para restringir su ámbito y dejar el espacio libre lo más pronto posible. Las variables de control de bucle son candidatas comunes para esta clase.

Las variables *estáticas* por lo general son locales a una función o bloque, pero difieren de las variables automáticas en que el almacenamiento asignado y el valor allí almacenado se conservan durante la ejecución de la función o el bloque. Un posible uso de una variable estática podría ser contar la cantidad de veces que se llama a una función. Asimismo, pueden proporcionar "ocultación de datos" por cuanto los valores almacenados en tales variables no son accesibles externamente, a diferencia de las variables externas. Una variable estática que se declara externamente a un juego de funciones está disponible globalmente para esas funciones, pero no está disponible para las funciones ajenas a ese archivo determinado.

La utilización de matrices

Una matriz se declara del mismo modo que otras variables, con su tamaño (es decir, la cantidad de elementos) entre corchetes a continuación del nombre de variable.

Por ejemplo:

```
int matrizent[100];
```

reserva espacio de almacenamiento para los elementos de matriz matrizent[0], matrizent[1], etc., hasta matrizent[99]. En C los subíndices siempre parten desde cero, y la declaración se refiere a la cantidad total de elementos, de modo que, en este caso, no hay ningún elemento matrizent[100]. Las matrices estáticas o externas se pueden inicializar mediante la adición a la declaración de una lista de valores encerrados entre llaves.

Por ejemplo:

```
static int días_del_mes[12]=
      {31,28,31,30,31,30,31,31,30,31,30,31};
```

Si la lista no está completa, los elementos restantes se establecen en cero. De no efectuarse ninguna inicialización para una matriz estática o externa, todos los elementos se inicializarán a cero. Las matrices automáticas no se pueden inicializar y el espacio se creará con valores de desecho, de modo que no se puede confiar en que los elementos se inicialicen en cero.

Obviamente, no tiene sentido la idea de una matriz de registro.

Si se está inicializando una matriz, el C no exige que se mencione el tamaño; éste se tomará automáticamente como la cantidad de valores proporcionados, de modo que la declaración anterior muy bien podría haberse escrito como:

```
static int días_del_mes[]=
      {31,28,31,30,31,30,31,31,30,31,30,31};
```



Esta facilidad es de especial utilidad para series, o matrices de tipo char, donde la serie inicializadora simplemente se puede encerrar entre comillas. De modo que las dos declaraciones siguientes son equivalentes:

```
static char st[]="hola";
```

y

```
static char st[]={'h','o','1','a'};
```

Observe, sin embargo, que estas series no son dinámicas en el mismo sentido que las series del BASIC: su longitud no puede variar respecto a aquella declarada.

En un ulterior capítulo veremos con mayor detalle la manipulación de series.

El C puede abordar matrices con virtualmente cualquier cantidad de dimensiones. El único punto a recordar respecto a las matrices de dos o más dimensiones es que cada subíndice debe tener su propio par de corchetes, de modo que la declaración para una matriz bidimensional de cuatro por cinco elementos se entraría como:

```
int matrizbid[4][5];
```

Las matrices pueden ser pasadas a las funciones como parámetros, pero en este caso se las pasa por referencia; es decir, se pasa a la función la dirección del primer elemento y cualquier cambio en la matriz realizado dentro de la función permanecerá vigente cuando se vuelva a transferir el control a la rutina de llamada. No es necesario declarar el tamaño de una matriz dentro de la función, porque de todos modos éste se conoce, de modo que las funciones se pueden escribir con un carácter bastante general para que operen sobre matrices de cualquier tamaño.

Los listados que ofrecemos como ejemplo emplean muchos de los conceptos analizados aquí. Estamos suponiendo que se requiere un generador de números aleatorios y las funciones para manipularlo están escritas en un archivo, el cual se puede unir a cualquier programa que necesite números aleatorios.

En un archivo separado hay un breve programa de prueba que requiere un conjunto grande de números aleatorios y cuenta las frecuencias de números en intervalos dados para comprobar una distribución uniforme.

Listado 1, almacenado en archivo 1

```
#define MULT 25173
#define MODULUS 65536
#define INCREMENT 13849
static int semilla;
/*esta declaración es externa por cuanto concierne a
este archivo de modo que la variable semilla está
disponible para todas las funciones de este archivo,
pero no se puede aludir a ella fuera del archivo; de
hecho el nombre "semilla" se puede usar libremente
en otros archivos*/
randomise(s)
int s
{
    semilla=s;
}
random(n)
int n
/*devuelve un número aleatorio entre 0 y n*/
{
    semilla=(MULT* semilla+INCREMENT)%
  MODULUS;
/*el método de congruencia lineal*/
    return((int)((double)semilla/(double)MODULUS*
  (double)n+0.5));
/*observe la conversión de los valores enteros a punto
  flotante para la aritmética y otra vez a int, el +0.5 es
  para redondear al entero más próximo*/
}
double rnd()
/*devuelve un número aleatorio entre 0 y 1*/
{
    semilla=(MULT* semilla *INCREMENT)%
    MODULUS;
    return ((double)semilla/(double)MODULUS);
}
```

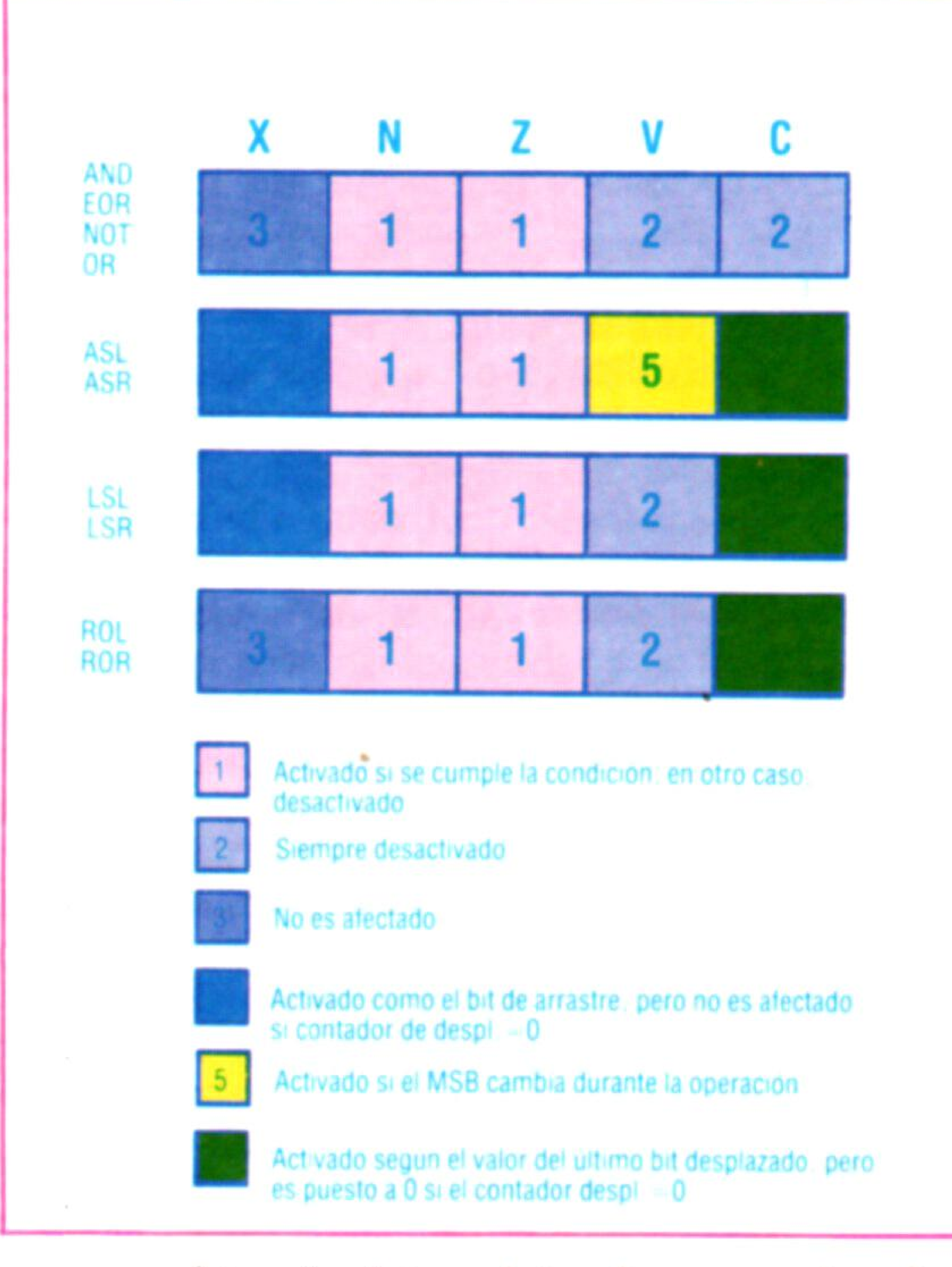
Listado 2, almacenado en archivo 2

```
#define SEMILLA 17
#define TAMAÑO 100
#define LIMITE 10000
#define NUM_DE_GRUPOS 10
int grupos []={10,20,30,40,50,60,70,80,90,100};
/*matriz externa se puede inicializar; ésta contiene
límites de grupo para contar frecuencias*/
int frecuencias [10];
/*esta matriz externa se inicializará en ceros*/
main()
int r;
{
    randomise (SEMILLA);
    register int i;
/*utilizando clase de registro para variables
de bucle para proceso más eficiente, y
guardando la declaración lo más tarde
posible*/
    for (i=0;i< LIMITE;++i)
    {
/*tomar 10000 números aleatorios entre 0 y 100*/
            r=random (TAMAÑO);
            register int j;
/*comprobar a qué grupo pertenecen*/
            for (j=0;j< NUM_DE_GRUPOS;
            ++j)
            {
            if (r < grupos [j])
            {
/*contar el número aleatorio en el grupo
correspondiente y salir del bucle */
                ++frecuencias [j];
                romper;
            }
        }
    }
/*imprimir una tabla de frecuencias para comprobar
distribución uniforme*/
      for (j=0;j< NUM_DE_GRUPOS;++j)
      {printf("%d_%d=%d\n",grupos [j]_
10,grupos[j],frecuencias[j])}
}
```

Rendimiento de los comandos

Nos corresponde analizar las instrucciones lógicas junto con las operaciones de rotación y desplazamiento y las instrucciones de bifurcación

Analicemos primeramente las instrucciones lógicas de que está provisto el 68000. La instrucción AND opera con un "Y" lógico el fuente con el destino, dejando el resultado en el destino y afectando en consonancia a los bits N y Z del registro de estado. Existen varios modos de direccionamiento posibles, pero el fuente o el destino por lo menos deben



Estado al día
Las instrucciones lógicas, de desplazamiento y de rotación afectan el contenido del registro de estado de la manera que aquí se ilustra. Las instrucciones de desplazamiento aritmético difieren de los desplazamientos lógicos sólo en el empleo que hacen del flag V para indicar cambios posibles en el bit de signo del operando

ser un registro de datos. Además se permiten los atributos de datos; así, por ejemplo, si D0=1010 1010 y D1=1111 0000, y ejecutamos la instrucción:

```
AND D1,D0
```

entonces resulta D0=1010 0000.

Éste es un sencillo ejemplo del empleo de una instrucción AND. En este caso, hemos tomado los cuatro bits menos significativos poniendo a cero los bits de máscara que están en D1, y hemos hecho que los MSB (bits más significativos) aparezcan en su estado original mediante una máscara de bits puestos a 1. Una versión más explícita de esta operación incluiría quizá la instrucción ANDI. Ésta acepta el modo de direccionamiento inmediato como operando fuente, y los modos alterables de datos como destino. Si empleamos ANDI en el ejemplo anterior tendríamos:

```
ANDI #$F0,D0
```

Las restantes instrucciones lógicas son OR y ORI para la operación OR lógica, EOR y EORI para la OR exclusiva, y la instrucción NOT. Estas instrucciones, respecto a los modos de direccionamiento, son parecidas a la instrucción AND, siendo idéntica la forma en que es afectado el código de condición. La manipulación de los bits con instrucciones lógicas es de gran importancia para controlar, por ejemplo, los bits que hacen de *flags*, o bien una palabra o las líneas de entrada/salida digitales para periféricos o equipos externos.

Cuando el operando de datos es un bit único, el 68000 tiene un conjunto de instrucciones para manipulación de cuatro bits que pueden ser utilizadas en lugar de las instrucciones lógicas. Estas instrucciones comprueban el estado de un bit específico (numerado de 0 a 31, en un registro de datos, sólo para palabras largas) o un byte de la memoria. Afectan también al bit Z del SR según sea el estado de ese bit. Por tanto, el bit Z se convierte en una memoria invertida de un bit respecto al bit especificado. La instrucción BTST puede usarse del modo siguiente. Si, por ejemplo, D0=XXXX XXXX XXX1 0000 (en binario, siendo X un 0 o un 1), entonces:

```
BTST #4,D0
```

comprobará el bit cuatro y pondrá Z a cero (el bit del ejemplo no es cero). Ahora bien:

```
BTST #3,D0
```

pondría Z a uno. Todos los códigos de condición permanecen inalterados.

Las restantes instrucciones afectan al bit que se comprueba. Y son:

```
BSET    comprueba y pone el bit a 1
BCLR    comprueba y pone el bit a 0
BCHG    comprueba y cambia el valor del bit
```

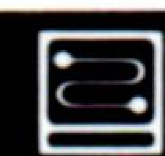
Así si ejecutamos:

```
BCHG    #4,D0
```

en el ejemplo anterior (donde D0 está puesto a 1) entonces D0 se convertiría en D0=XXXX XXXX XXX0 0000 y Z se pone a 0, que indica el estado antes del cambio.

Es de notar que todas estas instrucciones realizan las operaciones de comprobación y alteración de bits en una sola instrucción. Esto puede ser importante en un entorno de multiprogramación, donde la posibilidad de ser interrumpido entre dos instrucciones puede conducir a resultados impredecibles. Una observación final sobre las instrucciones lógicas es que aún si se aplica la comprobación, no es necesario realizar acción alguna a partir de esa información. Se puede emplear la instrucción como manipulador de bits.

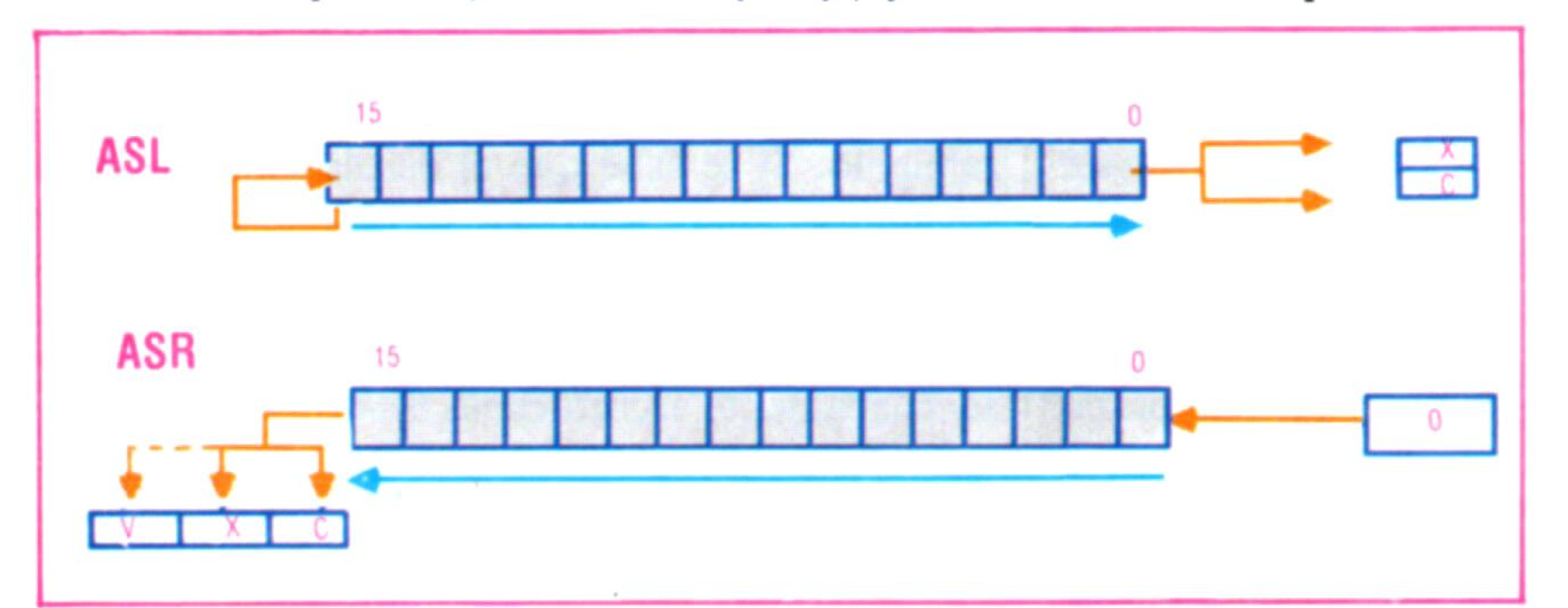
Ante todo, veamos lo que sucede cuando un patrón de bits es desplazado a la izquierda o a la derecha. Si D0 contiene 0000 0000 0000 1000, el desplazamiento de este modelo tres lugares a la derecha dará 0000 0000 0000 0001. El contenido de D0 ha sido dividido por ocho (o sea, 2^3), en otras palabras, un desplazamiento hacia la derecha corresponde a una división por dos por cada lugar que se desplace a la derecha. Además, se sigue de esto



que un desplazamiento a la izquierda equivale a una multiplicación por idéntica cantidad.

Obsérvese que, en nuestro ejemplo, hemos supuesto que se rellenan con ceros los nuevos dígitos binarios (por la izquierda en los desplazamientos a la derecha, y viceversa). Esto se deberá cambiar si deseamos conservar el signo de nuestro número cuando hacemos un desplazamiento a la derecha. Así, por ejemplo, si D0=1111 1111 1111 0000 (−16, en decimal) un desplazamiento a la derecha de tres lugares daría D0=1111 1111 1111 1110, que es −2 en decimal. Aquí hemos puesto unos por la derecha para poder mantener el signo negativo del número. En general, podemos decir que para desplazamientos a la derecha el signo se conserva introduciendo bits del mismo signo que el bit que indica el signo (en el operando de datos es el bit más significativo). Para desplazamientos a la izquierda, introduciremos siempre ceros en los bits menos significativos de la palabra. Si deseamos conservar el signo del operando de esta manera entonces la operación de desplazamiento se denomina *desplazamiento aritmético*.

En el 68000 estas instrucciones de desplazamiento aritmético son ASL para el desplazamiento aritmético a la izquierda (*Arithmetic Shift Left*) y ASR



Desplazamiento aritmético
Cuando se desplazan los bits a la derecha (ASR), la operación aritmética copia el bit 15 en el bit 14, pero el contenido del bit 15 permanece inalterado. El bit de signo se respeta. El bit 0 se copia tanto en el bit C como en el X del registro de estado (SR). Obsérvese que cuando se efectúa un desplazamiento a la izquierda, los ceros se copian en el bit 0, y el bit V se pone a 1 si el cont. del bit 15 ha cambiado

Desplazamiento lógico
Las operaciones de desplazamiento lógico se limitan a introducir ceros bien sea en el bit 15 bien en el bit 0, según que el desplazamiento sea a la izquierda o a la derecha. Pero debe tenerse en cuenta que V se pone a cero durante la LSL, y N se pone a cero durante la LSR. El último bit que se desplaza fuera se almacena en C y en X en ambas operaciones

para el desplazamiento aritmético a la derecha (*Arithmetic Shift Right*). Si el destino es un registro de datos, entonces podemos desplazar hasta ocho bits empleando el modo inmediato para el número de desplazamientos, o podemos servirnos del contenido de otro registro de datos como contador de desplazamientos hasta 31 bits en una palabra larga. Así, por ejemplo:

```
ASR     #8,D0
```

será el desplazamiento máximo en el modo inmediato, y:

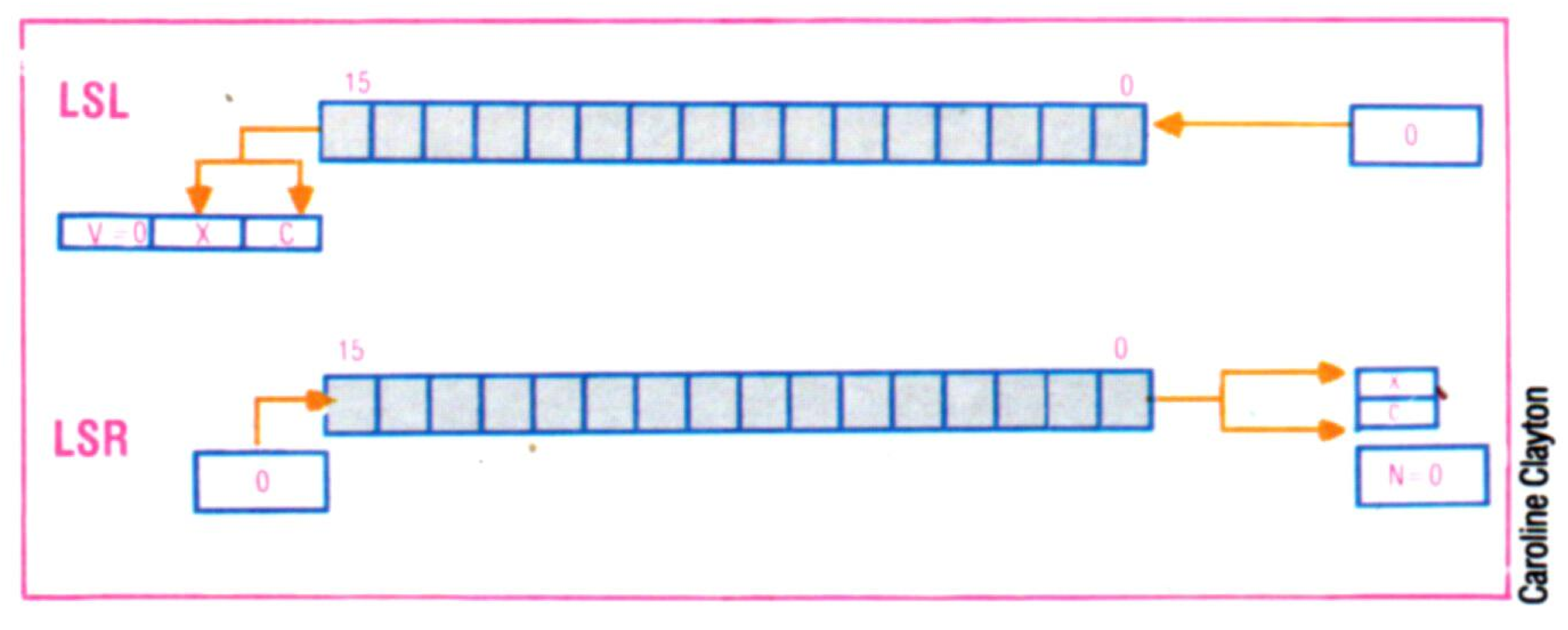
```
ASL     D1,D0
```

nos permitirá desplazar D0 a la izquierda un número de lugares indicado por el contenido de D1 (esto es, hasta 31 lugares). Si, no obstante, deseamos desplazar una posición de memoria, entonces la única opción posible será desplazar un lugar cada vez. Por ejemplo, ASR JUAN desplazará la posición de memoria indicada por JUAN un lugar a la derecha.

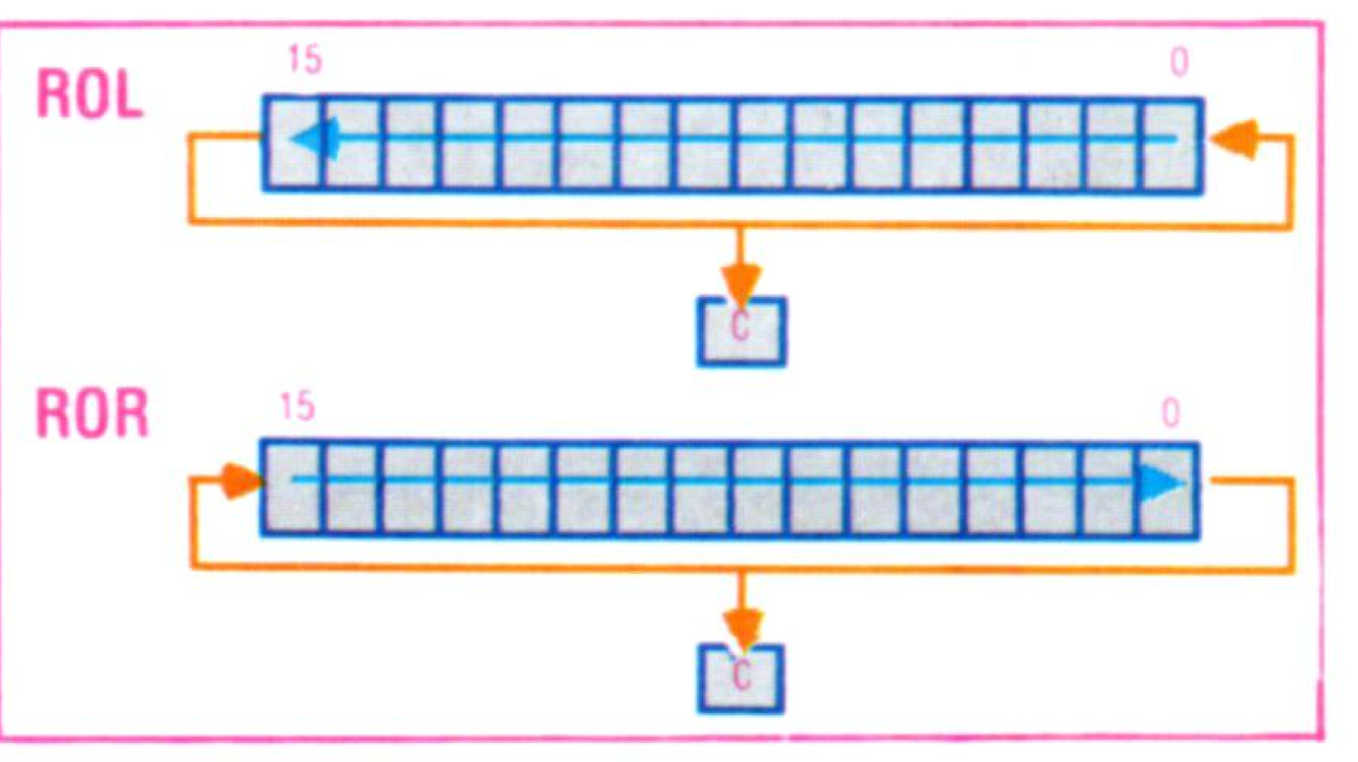
Se observará, por el dibujo del control de estado (página contigua), que se afectan todos los códigos de condición del SR (los bits C y X señalan los bits descartados por el desplazamiento, y el bit V indica cualquier cambio del bit de signo en desplazamientos a la izquierda). Los restantes bits, el N y el Z, reciben el valor acorde con el resultado.

Puede que usted se pregunte por qué se atribuye tanta importancia a estos desplazamientos aritméticos, considerando sobre todo que el 68000 está provisto de instrucciones para multiplicar y dividir. La razón está en que el tiempo de ejecución de estos dos tipos de operaciones es diferente. Una operación MULT tarda 70 ciclos de reloj y una ASL o ASR tan sólo tarda 6+2n ciclos, donde "n" es el número de desplazamientos ejecutados. Así, el tiempo de ejecución de un desplazamiento aritmético está entre los ocho ciclos para un solo desplazamiento y los 68 ciclos para un desplazamiento completo de 31 bits. De esta manera, para un reducido número de desplazamientos, las instrucciones de desplazamiento serán considerablemente más eficientes que las instrucciones de multiplicar o dividir. En el caso extremo de una instrucción de dividir por dos ¡se conseguiría una rapidez 19 veces mayor!

Obsérvese que también disponemos de instrucciones de *desplazamiento lógico*, LSL y LSR, que no conservan el signo del operando de datos e introducen siempre ceros en los nuevos dígitos binarios. Las restricciones de direccionamiento de los desplazamientos aritméticos son también válidas en los desplazamientos lógicos, y el bit V siempre se pone a cero. El desplazamiento lógico se utiliza provechosamente para establecer o comprobar operan-



Caroline Clayton



Instrucciones de rotación
Las instrucciones ROTATE sólo hacen rotar a los operandos, poniendo el bit 0 en el bit 15 durante ROR y viceversa durante ROL. Obsérvese que el bit "intercambiado" se copia en el bit C del registro de estado durante la operación

dos de datos que contengan subgrupos o campos más pequeños.

Finalmente en este conjunto de instrucciones de desplazamiento toca el turno al grupo de instrucciones rotatorias. Son sin duda una reliquia histórica,

dado que su principal finalidad es la comprobación de bits singulares de operandos de datos mediante la rotación y comprobación del estado resultante en el flag de condición. Naturalmente, en el 68000 disponemos de un conjunto completo de instrucciones para comprobar y modificar bits, y tardan casi el mismo tiempo que el conjunto de las rotatorias. Quizá pueda bastar aquí un sencillo ejemplo de rotación. ROR #3,D0 rotará el contenido de D0 tres lugares a la derecha, y dará al bit C un valor acorde con el último bit que ha dado la vuelta, yendo del menos significativo al más significativo.

Avancemos algo más para examinar las instrucciones de control de programas. Se trata de un importantísimo conjunto de instrucciones que controlan la secuencia de ejecución. Un grupo, denominado *bifurcaciones condicionales*, alterará el flujo secuencial normal de las instrucciones en función de una condición que se comprueba. Las *bifurcaciones incondicionales*, por su parte, siempre producirán una bifurcación o cambio del flujo normal secuencial de las instrucciones. Examinaremos primero estas últimas.

El modo habitual de ejecutar una bifurcación incondicional es:

```
BRA        NUEVAETIQ
```

donde el flujo de instrucciones continuará a partir de la posición denominada NUEVAETIQ una vez ejecutada la instrucción BRA. Si el desplazamiento de un byte puede guardarse en una palabra de ocho bits con signo, la instrucción resultante se codificará en una palabra. La mitad de la palabra contendrá el opcode BRA (60 en hexa) y el otro byte contendrá el desplazamiento de 16 bits con signo. Si el desplazamiento no es posible contenerlo de esta manera, o no se conoce todavía la dirección de bifurcación (es decir, el assembler no puede calcularla porque implica una referencia ulterior), entonces el byte de desplazamiento contendrá valor cero y la palabra siguiente contendrá el desplazamiento de 16 bits completo con signo.

Por lo general, la instrucción BRA será la adecuada; no obstante, en algunas ocasiones desearemos hacer algo más refinado cuando se calcula la dirección de la bifurcación. Por ejemplo, cuando se quiere bifurcar a una dirección contenida en una tabla con un índice establecido en un registro (recuérdese que esta manera de direccionar se denomina *indirecta con índice y desplazamiento*). Por desgracia, la instrucción BRA no acepta esta forma de dirección calculada, y así Motorola ha proporcionado la instrucción JMP (*jump*: saltar), donde es posible el cálculo de la dirección de bifurcación si es preciso.

Veamos un ejemplo de programa que ilustra las diferentes formas de bifurcación incondicional. La lista adjunta (*De un lugar a otro*) ilustra los principios en que se basa. Las instrucciones de salto en este ejemplo muestran un direccionamiento absoluto (hacia adelante y hacia atrás) e indirecto con indexación y desplazamiento. Otros modos de direccionamiento permitidos son el *indirecto simple* —por ejemplo, (A2)— y el *relativo al PC*.

Las instrucciones BRA muestran primero un ejemplo donde el desplazamiento de la dirección se contiene en la palabra de instrucción, y después dos ejemplos donde se utilizan palabras de extensión completa. En el primero de ellos (BRA FINISH) se ha utilizado una extensión completa de palabra aunque el desplazamiento puede ser contenido en un byte. Esto se debe a que el assembler no conoce la dirección de FINISH todavía, y por ello debe proveer una extensión completa de palabra para el desplazamiento. Si se sabe que el desplazamiento hacia adelante se ajusta a un byte con signo, entonces se puede obligar al assembler a que utilice la forma abreviada de la instrucción mediante el sufijo .S (*short*: breve). Así, nuestro ejemplo podría ser más adecuadamente escrito como BRA.S FINISH.

Examinemos ahora el segundo grupo de instrucciones de bifurcación: las *bifurcaciones condicionales*. Este grupo se subdivide en tres subgrupos:

- Bifurcaciones de complemento a dos
- Bifurcaciones sin signo
- Control de bucles

Los dos primeros subgrupos tienen un formato común en las instrucciones:

```
Bcc ETIQ
```

donde cc se refiere al código de condición que se comprueba. Si esta condición es verdadera, entonces tiene lugar una bifurcación a ETIQ; en caso contrario se ejecuta la siguiente instrucción de la secuencia. La condición comprobada se muestra en el cuadro *Condiciones para bifurcar* (derecha).

La columna "Verdadero si" del cuadro es la condición aritmética resultante de la comparación por medio de CMP o la instrucción SUB (que, obviamente, se ejecuta inmediatamente antes de aplicarse la bifurcación condicional). En el caso del primer subgrupo de condiciones mostrado en el cuadro (las bifurcaciones del complemento a dos), el bit V, o flag de desbordamiento, se incluye en todo caso en la comprobación lógica y éste es el factor que determina la pertenencia a este subgrupo. Implica además la necesidad de una comprobación adicional del bit de desbordamiento para una corrección completa, por lo que es preciso examinar cuidadosamente las condiciones lógicas para las bifurcaciones indicadas en el *Manual del usuario*. Por ejemplo, una bifurcación BGE comprueba si N=V, y habrá una bifurcación si se cumple NOT N AND NOT

De un lugar a otro

```
                    WAYOFF EQU     $3FFF
                           ORG     $1000
100     3200        START  MOVE    D0,D1
1002    4EF8 1000          JMP     START      *Extensión pal. completa
1006    4EF8 101C          JMP     FINISH     *De nuevo pero hacia adel.
100A    4EE8 0005          JMP     5(A0)      *Indirecto con desplaz.
100E    4EF2 000C          JMP     12(A2,D0   *Indirecto con índice y
                                               desplazamiento

    *Versiones que siempre bifurcan

    *Desplaz. en la palabra del opcode si está en un byte

1012    60EC               BRA START          *Desplazamiento negativo
1014    6000 0004          BRA FINISH         *Hacia adelante, sin saberse
                                               cuánto hacia adelante
1018    6000 2FE5          BRA WAYOFF         *Da una pal. de desplaz.
                                               de 16 bytes completa
101C    3200        FINISH MOVE D0,D1
```



V (es decir, si no hay desbordamiento y no es negativo), o cuando ambos N y V son verdaderos (desbordamiento y negativo).

Veamos un ejemplo. Supongamos que se desea comparar los dos números con signo contenidos en D1 y D2 y bifurcar hacia MAYOR si D2 es más grande que D1. Si la condición se comprueba antes con una instrucción comparativa, se emplea la bifurcación condicional BGT como sigue:

```
CMP   D1,D2        *forma D2-D1
BGT   D2MAYOR      *bifurca a D2MAYOR si no es
                   *cero ni negativo y no hay
                    desbordamiento
```

El segundo subgrupo de bifurcaciones condicionales ilustrado en el cuadro trata los números sin signo y la comparación de los códigos de condición es relativamente más sencilla. Por ejemplo, para comparar el contenido de la posición LUISA con D1, con independencia de cualquier condición de desbordamiento, se puede ejecutar esta secuencia:

```
CMP   LUISA,D1     *forma D1 - LUISA
BEQ   IGUAL        *bifurca a IGUAL si z=1
```

Un segundo ejemplo de estas bifurcaciones condicionales se da cuando se codifica en ensamblador para la realización de un bucle. Por ejemplo, el ensamblador equivalente de:

```
FORI:=1 TO 5 hacer
(...parte principal del programa a ejecutar 5 veces...)
NEXT I
```

sería:

```
      MOVEQ  #5,D7   *establece contador bucle
LOOP  (...parte princ.  prog. a ejecutar 5 veces...)

      SUBQ   #1,D7   *decrementa el contador
      BNE    LOOP    *repite hasta que D7=0
```

quedando codificado eficazmente en sólo tres palabras (dado que se han usado las instr. rápidas).

El tercer subgrupo de bifurcaciones condicionales está formado por una única instrucción, DBcc (*Decrement and Branch*: decrementar y bifurcar al cumplirse la condición cc). Esta instrucción es una ampliación del programa de control de bucles dado más arriba, pero codificando tanto el decremento como la bifurcación condicional en una sola instrucción. De hecho la instrucción imita el pseudo-code típico del PASCAL para el bucle reiterativo:

```
REPEAT
  (...cuerpo del bucle...)
UNTIL
  'c'=verdadero o Dn=-1
```

donde cc es una de las condiciones descritas en el segundo subgrupo ilustrado en el cuadro y Dn es un registro de datos empleado para retener el contador del bucle. Veamos cómo puede codificarse con la instrucción DBcc:

```
      MOVEQ #5,D1 *establece contador bucle
LOOP  (...cuerpo del programa...)
      DEBQ D1,LOOP *salida si el último 'cc' es 0
                   *o bien D1=1 (6 iteraciones)
```

Obsérvense con cuidado las condiciones de salida explicadas en los comentarios anteriores, y cómo además el sentido de la bifurcación es diferente respecto de la instrucción BEQ normal. Si la comprobación condicional no es necesaria, entonces puede emplearse una cc de F (de "falso") para dar el equivalente DBcc de un simple bucle con FOR. Por ejemplo:

```
      MOVEQ #4,D3
LOOP  (...cuerpo del bucle FOR...)
      DBF   D3,LOOP
```

Lo cual equivale a nuestro bucle FOR original de cinco iteraciones, y hasta ocupa la misma memoria (todas las instrucciones DBcc ocupan dos palabras). Algunos programadores opinarán, sin embargo, que nuestra codificación primera era más explícita.

Condiciones para bifurcar

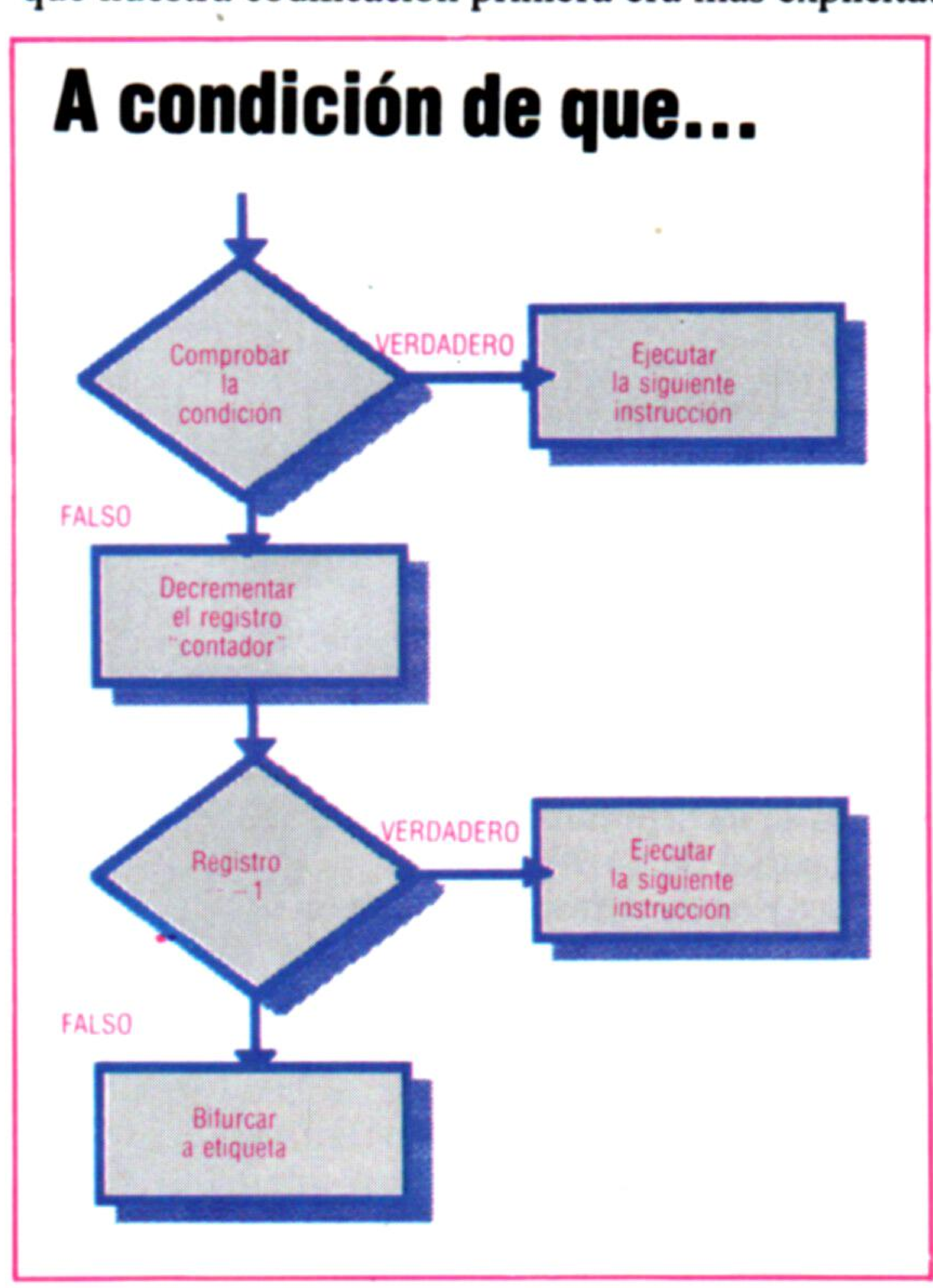
Las siguientes condiciones pueden comprobarse antes de una bifurc. hacia una dir. calculada por medio de la aritmética del complemento a dos

Condición cc	Verdadero si
GT mayor que	destino>fuente
LT menor que	destino<fuente
GE mayor o igual que	destino>=fuente
LE menor o igual que	destino<=fuente
VS desbordamiento	V=1
VC no hay desbord.	V=0

Las siguientes condiciones pueden comprobarse antes de realizar una bifurcación hacia una dirección calculada mediante enteros sin signo

EQ igual a cero	Z=1
NE no igual a cero	Z=0
MI menos	N=1
PL más	N=0
HI más alto que	C=Z=0
LS más bajo que	C o Z=1
CS arrastre activado	C=1
CC arrastre limpio	C=0

A condición de que...



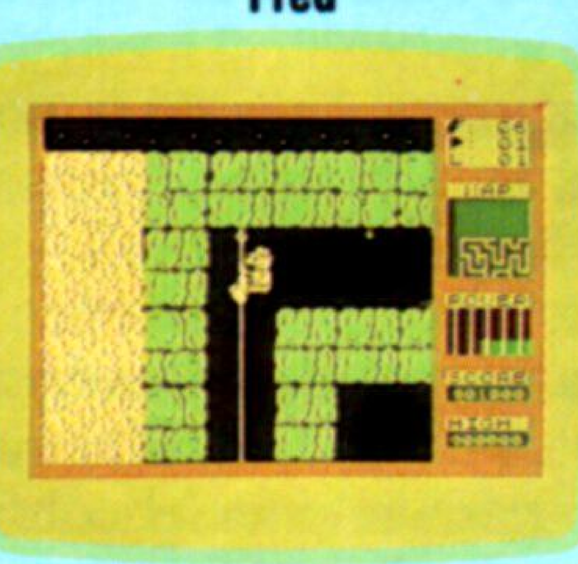
Tiempo para comprobar
La DBcc proporciona al programador del 68000 varias y poderosas oportunidades de programación a alto nivel. El diagrama de flujo de la instrucción es el que aquí se muestra. Obsérvese que no sólo se limita a decrementar el registro contador (compararla con la instrucción DJNZ del Z80), sino que también comprueba la condición. A veces, sin embargo, podemos necesitar la ejecución de un bloque entero de código reiteradamente (como en el bucle FOR...NEXT del BASIC), pudiéndose emplear para este fin la instrucción condicionada FALSE (DBF)

Somos el mundo

Ant attack



Fred



Gilligan's gold



Kokotoni Wilf



Sorcery



Visualizaciones cortesía de "Your Spectrum"

Show benéfico
Vemos aquí una selección del software correspondiente a la versión de SoftAid para el Spectrum. Los juegos escogidos para la compilación eran todos clásicos de estilo recreativo. Los royalties y los beneficios generados por las ventas de la cassette se han donado al Band-Aid Trust

SoftAid es una organización creada por un grupo de firmas de software con el fin de recaudar fondos destinados a paliar el hambre en África

Es así como, unificando sus recursos, SoftAid ha producido una cassette de compilación que ha logrado reunir miles de libras que se entregarán a la institución benéfica Band-Aid. Los esfuerzos de esta institución, dirigida por Bob Geldof, tuvieron eco en el mundo. Tras el éxito del disco single *Do they know it's Christmas?* a fines de 1984, y los conciertos Live Aid en Londres y Filadelfia (Estados Unidos) en julio de 1985, numerosas organizaciones de otros campos del mundo del entretenimiento han ofrecido sus servicios gratuitamente para recaudar fondos para dicha causa. Una de las iniciativas que tuvieron mayor aceptación provino de las casas de software de juegos, a las que a menudo les agrada compararse con la industria de la música *pop*. El resultado fue SoftAid, una serie de cintas para los micros Commodore y Spectrum que incluían algunos de los juegos de mayor éxito en el mercado.

La fuerza motriz de SoftAid ha sido el Guild of Software Houses (GOSH), asociación gremial de la industria de juegos por ordenador cuya función habitual es intentar desterrar la piratería de software. La directiva del gremio preguntó a cada uno de sus asociados si estaban dispuestos a contribuir con un juego al proyecto. La idea original era que las casas de software escribieran juegos nuevos para la compilación, pero luego se decidió que era más factible recopilar juegos cuya popularidad entre el público ya hubiera sido demostrada. De modo que estos juegos se incorporaron en un único paquete al precio especial de £4,99.

Las casas de software mostraron una predisposición tan favorable a colaborar con el proyecto, que GOSH se encontró con más juegos de los que podía incluir en una única cassette para cada una de las dos máquinas para el que se había preparado originalmente la compilación SofAid. En consecuencia, hubieron de rechazarse ofertas de juegos de empresas tan conocidas como Llamasoft y Software Projects. El software seleccionado incluía algunos de los juegos más populares de los últimos años, entre ellos el clásico *Ant attack*. Los juegos que se incluyeron en la cinta para el Spectrum son: *Spellbound*, de Beyond; *Starbike*, de The Edge; *Kokotoni Wilf*, de Elite; *The pyramid*, de Fantasy; *Horace goes ski-ing*, de Melbourne House; *Gilligan's gold*, de Ocean; *Ant attack*, de Quicksilva; *3D tank duel*, de Real Time; *Jack and the Beanstalk*, de Thor, y *Sorcery*, de Virgin. Para el Commodore 64 los juegos incluidos son: *Gumshoe*, de A'n'F; *Beam rider*, de Activision; *Star trader*, de Bug-Byte; *Gyropod*, de Taskset; *China miner*, de Interceptor; *Kokotoni Wilf*, de Elite; *Gilligan's gold*, de Ocean; *Fred*, de Quicksilva; *Falcon patrol*, de Virgin, y *Flak*, de US Gold. Durante el desarrollo del proyecto, GOSH trabajó en estrecha colaboración con el Band-Aid Trust y, como atractivo adicional, se añadió al principio de cada cassette el single de Band-Aid *Do they know it's Christmas?*, proporcionando así otro incentivo para adquirir la interesante compilación.

Asimismo, GOSH respetó la filosofía general de Band-Aid de solicitar a todos los participantes en la producción y distribución de la cassette que prestaran sus servicios de forma gratuita. No sólo los juegos se cedieron gratuitamente, sino que todas las personas implicadas renunciaron a cobrar los gastos de impresión, publicidad, reproducción de cintas, distribución y venta al público, teniéndose que pagar solamente el costo de los materiales. Las cintas se pusieron a la venta en la primavera de 1985; inmediatamente se colocaron en los primeros puestos de las listas de *best-sellers* para las dos máquinas en cuestión, y consiguieron recaudar muchos miles de libras para el Band-Aid Trust. Y todo ello a pesar de que muchos de los clientes que adquirieron las cintas probablemente ya tenían en sus colecciones particulares algunos de los juegos incluidos en la compilación.

SoftAid: Para el Sinclair Spectrum y el Commodore 64. (Se están preparando versiones para el BBC Micro y la gama Amstrad.)
Editado por: The Guild Of Software Houses, 12-13 Henrietta Street, London WC2, Gran Bretaña
Autores: Varios
Palancas de mando: Opcionales
Formato: Cassette

Editorial Delta, S.A.



10114
9 788485 822836